ROZPOZNÁVANIE FONÉM ČÍSIEL SLOVENSKÉHO JAZYKA NEURÓNOVOU SIEŤOU

VOJTECH SLOVIK

2007

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KATEDRA APLIKOVANEJ INFORMATIKY

ROZPOZNÁVANIE FONÉM ČÍSIEL SLOVENSKÉHO JAZYKA NEURÓNOVOU SIEŤOU

(diplomová práca)

VOJTECH SLOVIK

Študijný odbor: Informatika

Diplomový vedúci: RNDr. Marek Nagy

Bratislava, 2007

Čestne prehlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej literatúry a s odbornou pomocou diplomového vedúceho.

.....

Vojtech Slovik

POĎAKOVANIE

V prvom rade by som chcel vyjadriť vďaku môjmu diplomovému vedúcemu Marekovi Nagyovi, ktorý mi poradil hlavne v kľúčových momentoch tejto práce a na ktorého príjemných prednáškach o rozpoznávaní reči som prišiel na tému tejto práce.

Ďakujem všetkým ktorý mi prepožičali svoj hlas na vytváranie vzoriek a významne pomohli vytvoreniu práce, menovite Janke Ošustovej a Dušanovi Plašienkovi. Ďakujem rodine a najbližším, že v posledných mesiacoch vytvárania tejto práce to so mnou vydržali, tiež všetkým ktorí ma v tomto čase otravovali menej než zvyčajne.

Jánovi Žabkovi, že mi kedysi dávno vnukol myšlienku študovať umelú inteligenciu a „Matfyzu“ za to, že mi to dovolil zrealizovať.

Ďakujem aj všetkým tvorcom linuxových systémov, že ma postavili pred neuveriteľné a nečakané technické problémy, ktoré bolo neustále treba riešiť.

ABSTRAKT

Práca sa venuje rozpoznávaniu čísiel slovenského jazyka. Jednak na malom slovníku čísiel, ale okrajovo aj problému rozpoznávania prirodzených čísiel do milión. V práci bola použitá metóda klasifikovania pomocou neurónových sietí. Klasifikovanými kategóriami boli fonémy a práca venuje dosť veľa priestoru vytváraniu vzoriek foném. Tiež je tu popísaný spôsob ako integrovať Viterbiho alogritmus do procesu rozpoznávania a navrhnuté praktické vylepšenia tejto integrácie.

Kľúčové slová: rozpoznávanie reči, fonémy, neurónové siete, Viterbiho algoritmus

1 Úvod

Reč je najjednoduchším - najľahšie naučiteľným a komplexným prostriedkom komunikácie medzi ľuďmi. Taktiež môže byť aj vstupom v počítačovej aplikácii a zlepšiť či zjednodušiť komunikáciu smerom od človeka k prístroju. Preto je porozumenie hovorenej reči počítačom užitočným cieľom. Zatiaľ tento cieľ nie je dosiahnutý v takej miere akej je schopný človek.

Pod porozumením reči mám rozumieme viacfázový proces. Najhrubšie rozdelenie môže byť na:

1. digitalizáciu analógového signálu (transformácia do formy, ktorú je schopný analyzovať program v počítači)

2. rozpoznanie jednotlivých slov (priradenie slov ku častiam digitálnej vzorky)

3. pochopenie významu (syntaktická a sémantická analýza)

Prvú fázu sme schopní realizovať dostatočne presne. To znamená, že vieme rozkladať zvukový signál do frekvenčného spektra so zvolenou (ľubovoľnou) presnosťou. A teda vieme aproximovať proces, ktorý sa deje v slimáku ľudského ucha, kde sa rozkladá zvuk do frekvenčných pásiem 100 Hz – 20 kHz.

V druhých dvoch fázach sa dosahujú uspokojivé výsledky len v špecifických doménach, ale úspešnosť druhej fázy aspoň vieme objektívne merať. Najpodstatnejšími parametrami úspešnosti pri rozpoznávaní slov sú veľkosť slovníka a závislosť na rečníkovi. Pre konkrétneho rečníka sa dosahuje slušná úspešnosť (92%) rozpoznávania aj na veľkom slovníku (anglický jazyk), taktiež na malom slovníku (čísla) pre všeobecného rečníka je postačujúca úspešnosť (94%), ale vo všeobecnosti nie je problém rozpoznávania uspokojivo vyriešený.

Jedným z prístupov je rozpoznávať jednotlivé fonémy povedaného slova. Z týchto skladaním určiť o aké slovo sa jedná, pričom na klasifikovanie fonémy sa použije neurónová sieť. Výhodou tohto prístupu je, že slovník potenciálne rozpoznávaných slov sa dá ľahko rozšíriť. Nakoľko nie sú rozpoznávané celé slová, ale iba ich časti – fonémy. Pre anglický jazyk sa takto dosiahla úspešnosť 94% pre všeobecného rečníka na malom slovníku čísiel. Keďže podobná analýza pre slovenčinu neexistuje, pokúsim sa dopracovať k podobným výsledkom. Ako sekundárny cieľ sa pokúsim zistiť úspešnosť rozpoznávania prirodzených čísiel do milión.

2 Automatizované rozpoznávanie slov

V tejto kapitole priblížim proces rozpoznávania slov rozdelený na dve hlavné časti: extrakcia pozorovaní zo zvukovej vzorky a klasifikovanie vzorky na slovo zo slovníka pomocou troch rôznych prístupov. Na záver určím ciele a spôsob rozpoznávania reči v tejto práci.

2.1 Problém

Vstupnými dátami pre problém rozpoznávania slov sú pozorovania z danej analógovej zvukovej vzorky. Aké detailné budú tieto pozorovania určujú parametre digitalizácie.

Digitalizácia signálu: Výstupom digitalizácie, konkrétne pulznej kódovej modulácie (*Pulse Code Modulation - PCM*), analógového signálu, sú diskrétne hodnoty tlaku zvukovej vlny v čase. Podľa požadovanej presnosti volíme vzorkovaciu frekvenciu (*sample rate*) a presnosť kvantizácie vzorky (*sample format encoding*). Pre kvalitný záznam rečového signálu je to minimálne 8 kHz a 12 bitov na vzorku, dôvody sú v [PSU]. Teda sekundová vzorka zvuku by bola zachytená minimálne ako 96 kb informácia. Pre potreby rozpoznávania reči sa štandardne, a preto aj v tejto práci, používa frekvencia 16 kHz po 16 bitov, t.j. každých 62,5 μs je zachytených 16 bitovou informáciou (256 kb / sec).

Úlohou pre rozpoznávací systém je priraďovať na základe týchto pozorovaní zvukové vzorky k potenciálnym slovám. V prípade tejto práce sú slovami fonémy slovenského jazyka. Fonéma je najmenšia zvuková jednotka. V slovenčine ich je 52 [IVA]. Ich zoznam je v dodatku 1 (*fonémy slovenčiny*).

Prvou fázou klasifikovania je extrakcia pozorovaní. Hodnoty z grafu zvukovej vlny totiž nie sú vhodné pre druhú fázu, ktorou je porovnávanie so slovníkom rozpoznávaných slov. Rôzne realizácie slovníka determinujú aj spôsob porovnávania.

2.2 Extrakcia pozorovaní - frekvenčná analýza

Rozklad na frekvencie: Aby sa pri rozpoznávaní spracovávali signifikantné parametre signálu, tak sa používajú metódy uvažujúce jeho rozklad na frekvencie. Takto dochádza k lepšej

aproximácii činnosti ľudského ucha, a taktiež nie je proces porovnávania spomaľovaný analýzou redundantných údajov.

Za v praxi nutného predpokladu približnej stacionarity signálu môžeme analyzovať signál v jednotlivých krátkych (rádovo desiatky milisekúnd) časových úsekoch tzv. okienkach (*frames*). Pre každé z týchto je možné vypočítať vektor príznakov. Väčšina metód je založená na Fourierovej transformácii, keďže jej výstupom je graf zastúpenia frekvencií v čase. V praxi sa implementuje jej rýchlejšia verzia *Fast Fourier Transform* (FFT). S jej využitím sa používajú dva dôležité prístupy:

1. Mel-frekvenčné kepstrálne koeficienty (*Mel Frequency Cepstral Coefficients - MFCC*) Zlogaritmovaním absolútnej hodnoty Fourierovej transformácie a výpočtom inverznej Fourierovej transformácie dostávame Kepstrálne koeficienty, ktoré sa upravia podľa *mel škály*[1]. Posledná úprava sa robí kvôli tomu, že závislosť medzi výškou tónu ako ho vníma ucho a jeho frekvenciou nie je lineárna.

2. Lineárne prediktívne kódovanie (*Linear Predictive Coding - LPC*) odhaduje parametre modelu vytvárania reči priamo z rečového signálu. Signál sa popisuje lineárnou kombináciou predošlých vzoriek, pričom koeficienty systému rovníc, ktorý modeluje hlasový trakt, sa vypočítajú iteratívnym algoritmom. Podobne ako pri prvom prístupe je možné vypočítať aj kepstrálne LPC koeficienty (*cepstral envelope*).

Tieto prístupy, a ďalšie ako energia signálu[2] a prechody nulou, sú podrobne popísané v [PSU] a algoritmy na výpočet spomínaných koeficientov sú implementované v [HTK]. Po aplikovaní niektorej z týchto krátkodobých frekvenčných analýz, máme efektívne a s presnosťou akou potrebujeme popísané jednotlivé časové okienka. A to pomerne biologicky plauzibilne, keďže membrána ľudského ucha je rozdelená na oblasti, ktoré reagujú na danú frekvenciu signálu. Detaily psycho-fyzikálnych aspektov sú v [NAG].

Ďalší postup už závisí od slovníka, v ktorom sú slová, ktoré môžu byť rozpoznané. A to najmä od jeho veľkosti a dĺžky slov v ňom. Existujú tri základné modely klasifikátorov slov, ktoré boli porovnávané aj v [OND]. Odlišujú sa v spôsobe porovnávania spočítaných vektorov príznakov klasifikovaného slova so slovami zo slovníka. Tu len v krátkosti spomeniem ich princípy a dosiahnuté výsledky.

---

[1] logaritmická škála frekvencií založená na ľudskej percepcii výšok tónov m = 1127.01048 ln (1 + f / 700)
kde m – mely, f - frekvencia

[2] súčet plochy pod a nad krivkou signálu

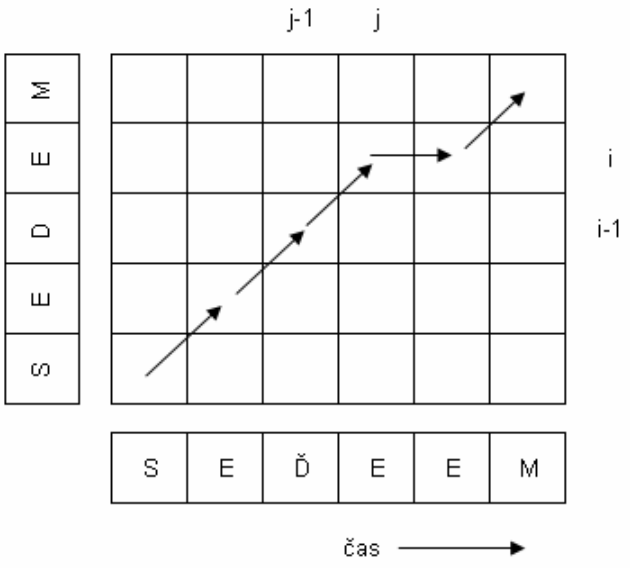
2.3 Modely klasifikátorov

2.3.1 Dynamické krivenie časovej osi

(*Dynamic Time Warping – DTW*)

DTW je metóda založená na porovnávaní vzoriek príznakov tak, že ich vnútorne časovo naťahuje a hľadá najlepšiu zhodu k vzorke zo slovníka (vzhľadom na nejakú metriku) [obr 1].

V slovníku sa nachádzajú „celé ľudské“ slová, pričom každé je popísané postupnosťou príznakových vektorov pre jednotlivé časové okienka $\mathbf{s_1, s_2, ... s_n}$. Na obrázku zodpovedajú časovým okienkam pre ilustráciu hlásky. Zvuková vzorka, ktorú chceme klasifikovať, je tiež popísaná nejakými vektormi príznakov $\mathbf{v_1, v_2, ... v_m}$. Vektory vzorky sa postupne porovnajú so všetkými slovami zo slovníka, pričom na základe metriky D(s,v) -> $\mathbf{R}$ [3] sa vyhodnotia jednotlivé vzdialenosti (podobnosti) vzorky od potenciálneho slova. Za rozpoznané slovo sa považuje to, ktoré dosiahlo minimálnu vzdialenosť od vzorky (najväčšiu podobnosť).



obr 1

Metrika na podobnosť slov sa dá implementovať jednoduchým dynamickým algoritmom, keďže je definovaná nasledovne:

[3] **s** a **v** sú postupnosti vektorov príznakov v čase.

$$D(s,v) = \min_{\{i(k),j(k),k\}} \sum_{k=1}^{K} \frac{d(i(k),j(k))W(k)}{N(W)}$$

i(k) (resp. *j(k)*) je funkcia výberu z postupnosti vektorov **s** (resp. **v**), pričom *k* je dĺžka spoločnej podpostupnosti indexov z **s** a **v**. *d* je metrika (napr. euklidovská vzdialenosť) na určenie vzdialenosti dvoch vektorov (jeden zo zvukovej vzorky a druhý z potenciálneho slova). *W* je váhovacia funkcia, ktorá zabezpečuje zmysluplný výber podpostupnosti, t.j. že sa vo výbere nebude príliš často opakovať ten istý vektor pozorovaní. Výber z postupností pozorovaní totiž realizuje vnútorné časové naťahovanie slov. *N* je normalizačný faktor, potrebný kvôli váhovaniu.

Záver z [OND] je, že DTW je vhodné na rozpoznávanie izolovaných slov. Pre 14 slovný slovník bola dosiahnutá priemerná 94% úspešnosť, a to nezávisle od rečníka. Čo je vynikajúci výsledok, ale táto metóda nie je vhodná pre väčšie slovníky. Využíva sa napríklad pri „hlasovom vytáčaní“ na mobilných telefónoch.

2.3.2 Skryté Markovovské modely

(*Hidden Markov models – HMM*)

Uvažujme model vytvárania reči, v ktorom sa počas vyslovovania hlasový trakt nachádza v rôznych stavoch. Tieto stavy reprezentujú artikulačnú konfiguráciu, ktorú dosahuje trakt na krátky časový úsek. Počas neho sa vygeneruje signál opísateľný spomenutými frekvenčnými analýzami. Keďže počet možných stavov sa dá ohraničiť na nejaký postačujúci konečný počet, tak môžeme využiť HMM na modelovanie priebehu reči ako náhodného parametrického procesu meniaceho sa v čase.

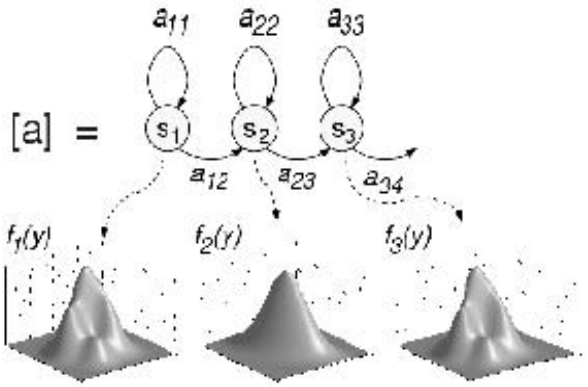
HMM v každom stave vygeneruje nejaký symbol z abecedy. Abeceda v princípe obsahuje (realizované pomocou kódovej knihy) všetky možné pozorovania, resp. signály. Tieto spektrálne vzory sa generujú na základe náhodných funkcií, ktoré ich pravdepodobnostne ohodnocujú pre každý stav. Keďže stavov aj signálov je konečný počet, sú tieto funkcie implementované pomocou matice pravdepodobností. Okrem tejto, je potrebná aj ďalšia matica, ktorá obsahuje pravdepodobnosti prechodov medzi stavmi[4]. Ostávajúcim parametrom HMM je vektor, ktorý obsahuje pravdepodobnosti pre stavy, že automat HMM v nich začne výpočet [obr 2].

Uvedené parametre modelov HMM (počet stavov, abeceda, matice a vektor pravdepodbností) v podstate realizujú slovník rozpoznávaných slov, pričom pre každú použitú fonému, resp. nejakú časť slova, je jeden HMM a ich spájaním modelujeme „celé ľudské“ slová alebo vety. Takýto

[4] keďže HMM tu modeluje vytváranie reči, tak pravdepodobnosti prechodov do stavu s nižším indexom sú nulové

model sa dá natrénovať napr. Baum-Welchovou metódou iteratívnej aktualizácie parametrov, popísanej v [PSU].

Pri rozpoznávaní treba určiť, ktorý model by s najväčšou pravdepodobnosťou vygeneroval rozpoznávané slovo. To je reprezentované ako postupnosť nejakých pozorovaní o_1 ... o_t (čo sú symboly abecedy HMM). Rekurzívne je možné vypočítať pravdepodobnosť, že model vygeneroval pozorovania o_i ... o_j, pričom začal (resp. skončil) v i-tom stave (resp. v j-tom stave). A teda vieme určiť aj s akou pravdepodobnosťou by HMM vygeneroval celú zadanú postupnosť pozorovaní[5].



obr 2

HMM pre fonému „a“

abecedou sú subfonémy a_{11} ... a_{34} ... , obsahuje stavy s_1, s_2, s_3 ... a pre každý z nich funkciu f_i, ktorá udáva, s akou pravdepodobnosťou by sa v danom stave vygeneroval nejaký spektrálny vzor a_{jk}

Už idea tohto prístupu napovedá, že bude vhodný pre rozpoznávanie aj pri väčšom množstve „ľudských“ slov, ktoré je možné poskladať z menších segmentačných jednotiek, napr. foném. V práci [SEM] bola dosiahnutá 84% úspešnosť na 16 vetách (s desiatkami slov) nezávisle od rečníka. HMM prístup je pravdepodobne využívaný aj v úspešných komerčných produktoch (napr. ScanSoft – *Dragon Naturaly Speaking*, IBM – *Via Voice*, MS Office – *Dictation*), kde sú dosahované nasledovné približné výsledky:

99% úspešnosť na malom slovníku (napr. čísla), natrénované pre jedného rečníka

90% – 95% pre veľký slovník (tisícky slov) pre jedného rečníka

[5] používa sa algoritmus Forward-Backward, popísaný v [PSU]

90% úspešnosť, ak rozpoznávanie nezávisí od rečníka (napr čísla).

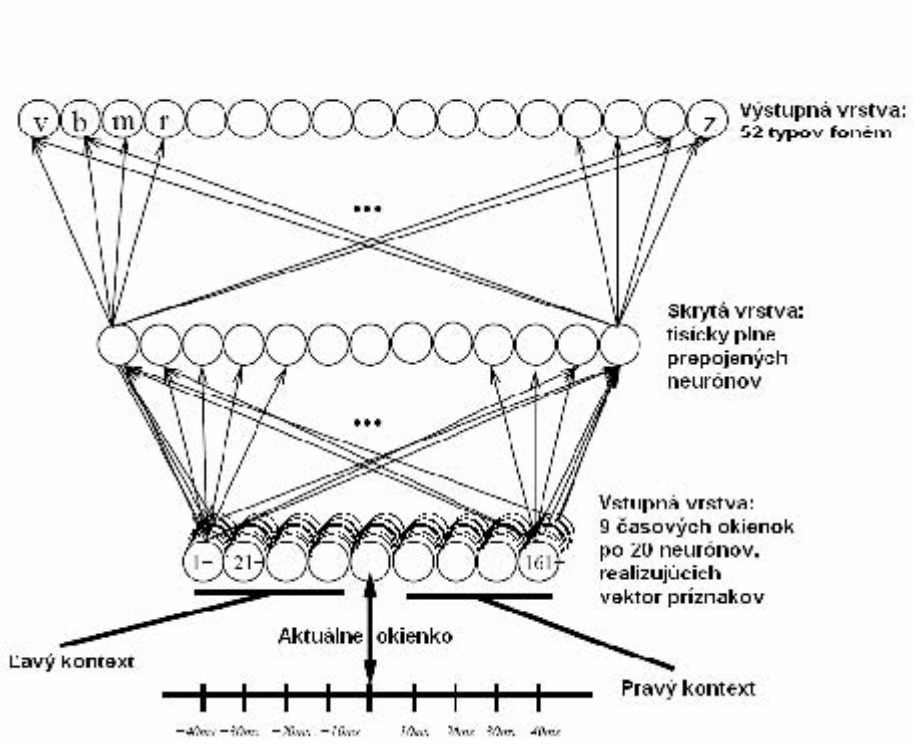
85% úspešnosť sa mi podarila odskúšať aj s voľne dostupným [HTK] (ako školský projekt na predmet Rozpoznávanie reči) – nezávisle od rečníka, avšak (z technických príčin) len na malom slovníku (26 slov).

2.3.3 Neurónové siete

(*Artificial neural networks -ANN*)

ANN majú schopnosť klasifikovať resp. zaradiť signál na vstupe do najpodobnejšej triedy, ktorá je popísaná nejakými vektormi príznakov. Navyše sú schopné sa naučiť ľubovoľnú spojitú funkciu, ktorou je aj vzorka hlasového signálu. Slovník klasifikovaných slov je teda v prípade ANN skrytý vo váhach natrénovanej siete. Trénovanie je prevedené ako učenie s učiteľom (*supervised learning*) pomocou algoritmu spätného šírenia chyby (*backpropagation algorithm*) – detaily v [UNS].

Nevýhodou štandardných klasifikačných neurónových sietí je, že nezahrňujú časový aspekt reči. To je však možné odstrániť modelmi s kontextovými neurónmi [obr 3].



obr 3

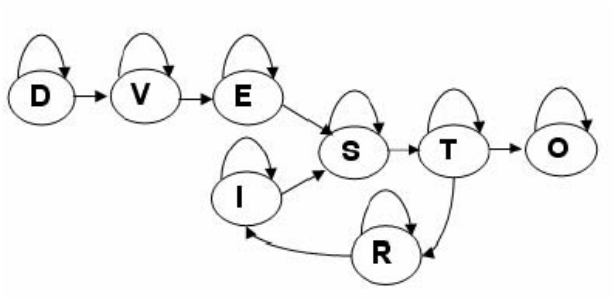
príklad klasifikátora foném s kontextovými neurónmi (ANN použitá v [MBA])

Taktiež možno porovnávať dostatočne krátke segmenty „ľudských slov“ (fonémy, alebo časti foném), čo odstráni problém, že klasifikátor založený na ANN má problémy s väčším množstvom tried. Z rozpoznaných segmentov je možné v záverečnej fáze poskladať „celé ľudské“ slová.

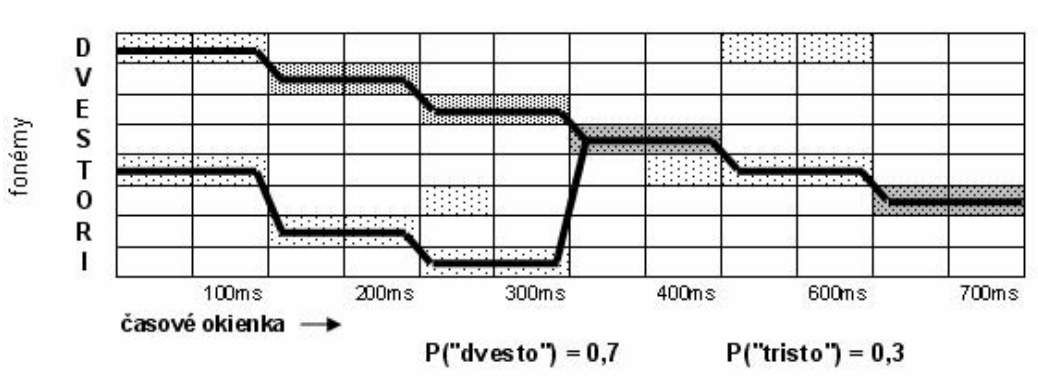
Viterbiho dekodér

(*Viterbi search*)

Výstupom ANN s lokalistickým kódovaním sú hodnoty pravdepodobností rozpoznania jednotlivých foném, a to pre každé časové okienko[6]. Vstupom do Viterbiho algoritmu je teda matica s pravdepodobnosťami jednotlivých foném v čase. Dekodér dynamicky nájde najodpovedajúcejšie slovo pre postupnosť pravdepodobností foném, berúc do úvahy gramatiku prípustných slov [obr 4]. Tá je pre potreby algoritmu uložená ako matica prechodov medzi fonémami.



obr 4
Príklad gramatiky dvoch prípustných slov



obr 5
Ukážka práce Viterbiho algoritmu na gramatike z [obr 4] a na matici pravdepodobností rozpoznaných foném v časových okienkach

[6] Čím je väčšia podobnosť signálu s nejakou vzorovou kategóriou fonémy, tým väčšia pravdepodobnosť na výstupe

Pseudokód algoritmu[7] podľa práce [RAB]:

N počet rozpoznávaných kategórií,
T počet časových okienok,
B[n][t], $1 \le n \le N$, $1 \le t \le T$ matica pravdepodobností foném v čase (výstup ANN),
A[i][j], $1 \le i \le N$, $1 \le n \le N$ matica prechodov medzi fonémami
π[i], $1 \le i \le N$ (apriórna) pravdepodobnosť že i-ta fonéma je prvou

hlavný cyklus vypočíta maticu δ - skóre pravdepodobnostne najlepších prechodov medzi fonémami a k nim korešpondujúce indexy jednotlivých kategórií foném do matice Ψ

δ [i][1] := π [i] * B[i][1], pre $1 \le i \le N$
Ψ [i][1] := 0, pre $1 \le i \le N$

```
pre všetky okienka t od 2 po T
        pre všetky fonémy n od 1 po N
                max_score := - ∞
                max_index := 0
                pre všetky fonémy i od 1 po N
                        ak (δ [i][t-1] * A[i][j] > max_score)
                                max_score := δ [i][t-1] * A[i][j]
                                max_index := i
                δ [j][t] := max_score * B[j][t]
                Ψ[j][t] := max_index
```

nájdenie víťaznej sekvencie foném a uloženie ich indexov do poľa fonemy

```
max_score := - ∞
max_index := 0
pre všetky fonémy i od 1 po N
        ak (δ [i][T] > max_score)
                max_score := δ[i][T]
                max_index := i
fonemy[T] := max_index

pre všetky okienka t od T-1 do 1
        fonemy[t] := Ψ [fonemy[t+1]][t+1]
```

Výsledky tohto prístupu[8] sú porovnateľne úspešné ako HMM: 92% úspešnosť na malom slovníku (čísla) nameraná nezávisle od rečníka (podľa [MBA] a [ASR]).

---

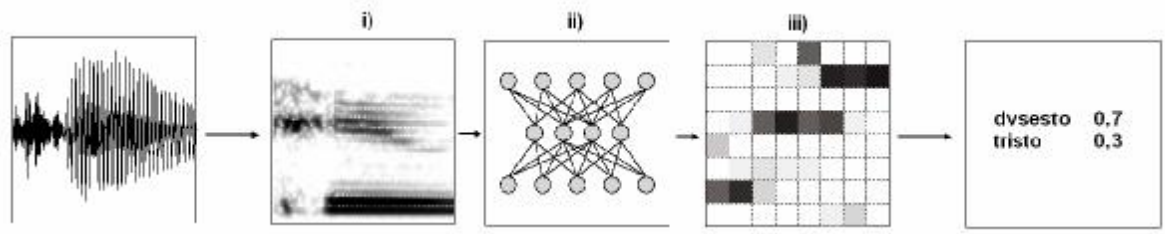
[7] v praxi sa implementuje v logaritmickej doméne čísel (so sčitovaním namiesto násobenia), kvôli problémom s reprezentáciou malých desatinných čísel

[8] Technika rozpoznávania pomocou ANN s Viterbiho algoritmom v ďalšej fáze je známa ako Hybridný HMM/ANN model, keďže technika gramatiky a Viterbiho dekodéra sa používa aj v HMM prístupe

2.4 Problémy pri klasifikovaní pomocou neurónových sietí

Pre anglický jazyk teda existujú pomerne kvalitné rozpoznávače slov závislé na rečníkovi a pre ľubovoľného rečníka sú dosiahnuté slušné výsledky na menších slovníkoch. Pre slovenčinu existujú len rozpoznávače pre špecifické využitie na malých slovníkoch (integrované do nejakého programu). Rozhodol som sa analyzovať problém rozpoznávania foném slovenského jazyka obsiahnutých v prirodzených číslach do milión nezávisle od rečníka, pretože v prípade dobrých výsledkov, by bolo možné rozšíriť úspešný model na väčší slovník. Z uvedených troch modelov klasifikátorov som zvolil prístup pomocou ANN. DTW má totiž problémy s väčšími slovníkmi a HMM je síce úspešné a univerzálne, ale predpokladám, že hľadanie optimálneho nastavenia pre slovenčinu v tomto prístupe, má väčší pomer parametre vs. architektúra, než je to u ANN. Navyše vhodnému nastaveniu parametrov pre rozpoznávanie slovenčiny v HMM bol venovaný priestor v [SEM]. Posledným dôvodom je, že ANN sa javí byť biologicky plauzibilnejší.

Hlavným cieľom mojej práce je nájsť vhodnú konkrétnu ANN architektúru a jej parametre pre rozpoznávanie foném v slovách slovenského jazyka pre univerzálneho rečníka. Najskôr sa pokúsim vyriešiť redukovaný problém: kvalitné rozpoznávanie pre jedného rečníka na fonémach obsiahnutých v slovenských číslach a následne dosiahnuť výsledky porovnateľné s prácami [MBA] a [ASR] (rozpoznávanie slovenských čísiel pre univerzálneho rečníka). Na sformulovanie podcieľov využijem popis jednotlivých fáz procesu rozpoznávania pomocou ANN:



obr 6

Fázy rozpoznávania: [obr 6]

- **i)** Frekvenčná analýza
- **ii)** Klasifikácia foném neurónovou sieťou
- **iii)** Nájdenie najvhodnejšieho slova zodpovedajúceho rozpoznaným fonémam

Vstupom do prvej fázy je audio signál vo formáte wav (*wave form audio format*). (Prvým malým technickým problémom je teda zvoliť parametre signálu.) Tento treba rozdeliť na okienka, na ktorých prebehne samotná frekvenčná analýza. Úlohou je teda aj zvoliť vhodnú časovú dĺžku okienka (cca. 5ms – 40ms) a tiež typ. Používa sa Pravouhlé okienko s rovnocennými váhami pre všetky hodnoty v ňom, alebo prekrývajúce sa Hamingove okienka s nižšími váhami na okrajoch. Ďalej treba zistiť ako čo najvhodnejšie počítať vektor príznakov (MFCC, LPC) a optimálnu dĺžku tohto vektora (cca. 10ky príznakov). Voľne prístupná parametrizovateľná implementácia je súčasťou [HTK].

Vstupom do druhej fázy sú vektory príznakov pre jednotlivé okienka. Použitá bude viacvrstvová dopredná sieť s kontextovými neurónmi na vstupe a s lokalistickým kódovaním na výstupe. Kvôli závislosti od času sa na vstup ANN pošle niekoľko okienok. Bude teda treba zvoliť vhodne dlhý kontext, taktiež počty neurónov v skrytých vrstvách a najmenší rozpoznávaný zvukový segment (fonéma, alebo jej časť).

Posledná fáza analyzuje maticu pravdepodobností foném v časových okienkach pomocou Viterbiho dekodéra. Ten využíva gramatiku rozpoznávaných „ľudských“ slov. Čiže okrem implementácie dekodéra je potrebné zrealizovať vhodnú transformáciu gramatiky do matice pravdepodobností prechodov medzi fonémami.

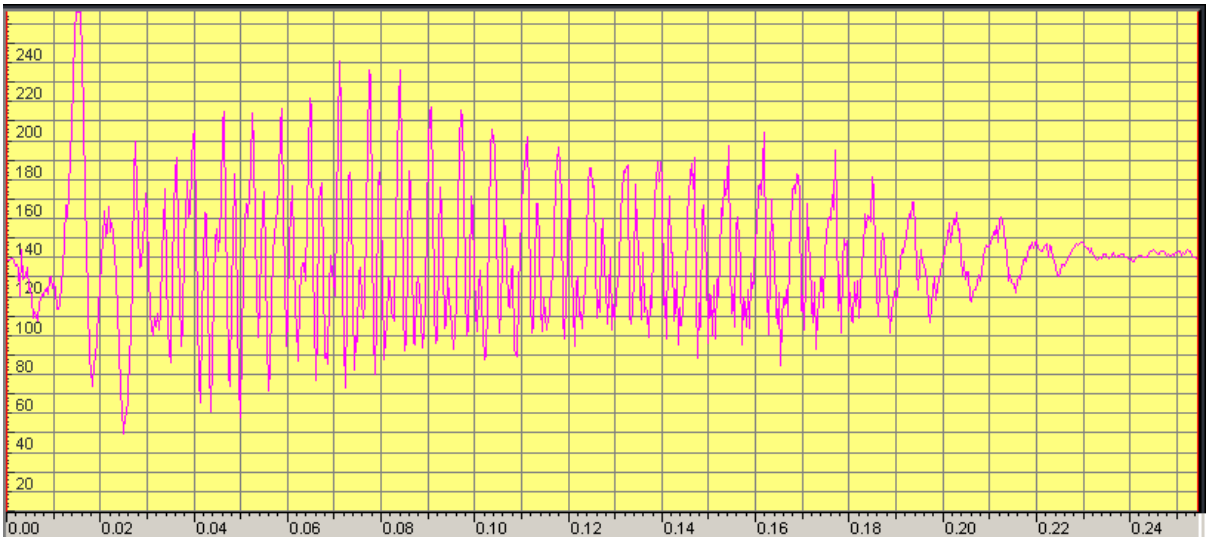
3 Fonémy ako vzorky

V tejto kapitole navrhnem spôsoby ako vytvárať vzorky foném. Popíšem ako presne boli vzorky vytvárané v rôznych prístupoch a aké problémy sa pritom vyskytli. Budem sa venovať aj problému výberu súboru základných zvukových jednotiek, ktoré bude klasifikovať neurónová sieť. Zhrniem poznatky o zvukových vlnách, ktoré som získal pri označovaní vzoriek.

3.1 Fonémy z neexistujúcich slov

Predpokladajme, že slová môžeme rozdeľovať na fonémy, podobne ako napísané slovo delíme na jednotlivé grafémy. Teda, že existuje množina foném **F**, taká že ľubovoľné slovo je postupnosťou foném z tejto množiny. Tento predpoklad sa zdá byť veľmi reálny, keďže pri počúvaní hovoreného slova máme dojem, že za sebou prichádzajú zvuky, ku ktorým vieme prideliť grafémy z abecedy nášho jazyka. A ku grafémam zase vieme priradiť fonémy – ako je vidno v dodatku 1 (transkripcia čísiel).

Na kvalitné natrénovanie nejakej fonémy nestačí spraviť nahrávku samotnej fonémy, pretože nie je ľahké ju verne vysloviť osamotene. Týka sa to hlavne spoluhlások, pretože keď sa snažíme vysloviť napr. samostatné „p“, tak vo zvukovej vlne, ktorú vytvoríme sa za vlnou zodpovedajúcou „p” nachádza ešte vlna podobná vlne fonémy „ä“ ako vidno na [obr 7] .



obr 7
vlna fonémy „p“ sa nachádza len po 30 milisekúnd (čas na vodorovnej osi je uvádzaný v sekundách)

Pochopiteľne, nie vždy vyslovíme danú fonému rovnako. Je to ovplyvnené rôznymi faktormi. Napríklad ráno máme nastavený hlasový trakt inak než večer. Taktiež nie v každom slove je fonéma vyslovená rovnako, napr. fonéma „ts“ v slove dvadsať (ds čítame ako c) je jemne odlišná ako fonéma „ts“ v slove tisíc.

Kvôli týmto faktom je výhodné aby sa fonémy použité ako vzorky nachádzali uprostred krátkeho slova. Za predpokladu, že existuje nejaký prototyp fonémy, ktorý sa vyskytuje (s malými rozdielmi) v slovách, je najlepšie zvoliť ako vzorky neexistujúceho slová. Napríklad pre fonému „p“ to môže byť slovo „apa“. Pre neurónovú sieť je potrebné mať dané slovo pooznačované[9] (sieť musí mať informáciu kedy začína a končí relevantný signál fonémy „p“).

Okrem nahrávania vzoriek foném (od čo najviac rečníkov), je teda potrebné manuálne označovať vzorky. Takto označovať všetky nahrávky zaberá veľa času, preto som sa rozhodol urýchliť proces označovania nasledovne: Pre každú fonému zavediem obaľovacie fonémy[10] a odhadnem priemerné hodnoty kedy začína a končí pozorovaná fonéma v jej nahrávkach. Tento odhad zrejme nebude dostatočne presný, preto som vytvoril aj pomocnú „označovaciu“ ANN, ktorú natrénujem na odhadom - pooznačované vzorky, a na tejto potom spustím klasifikovanie na tých istých vzorkách. Z výsledkov tohto klasifikovania sa automatizovane určia nové presnejšie hranice relevantnej fonémy. A na takto pooznačovaných vzorkách sa natrénuje hlavná „klasifikačná“ ANN.

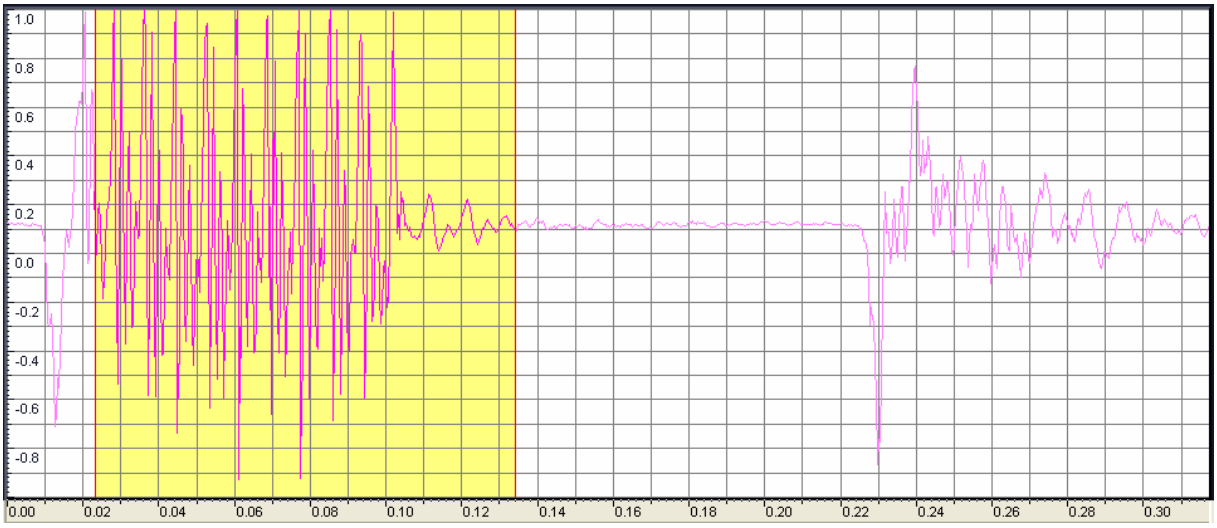
Pri vyslovovaní čísiel sa používa množina pozostávajúca len z 24 foném, ich zoznam je v dodatku 1 (*súbor foném 1*). Na takto vytváraných vzorkách foném som previedol prvý experiment rozpoznávania čísiel na jednom rečníkovi a ukázalo sa, že boli nepostačujúce. Výsledky sú uvedené v kapitole Experimenty – súbor foném 1.

3.2 Problémy pri vytváraní vzoriek foném

Predpoklad, že slová môžeme deliť na diskrétne úseky foném, a že existujú nejaké ich prototypy nie je celkom správny. Pri vytváraní vzoriek sa ukázalo, že v rámci slova sa pri niektorých fonémach nedá jednoznačne určiť kedy začínajú a končia. Týka sa to najmä spoluhlások. Fonémy, ktoré sa používajú pri vyslovovaní čísiel som rozdelil do 3 skupín podľa obtiažnosti hľadania ich hraníc v slove[11]:

ľahké (dlhé):	a, á, ä, e, i, í ia, m, n, o, ó s, š, u
krátke:	c, d, ď, p, r, t, ť, v
ťažké (krátke):	j, l

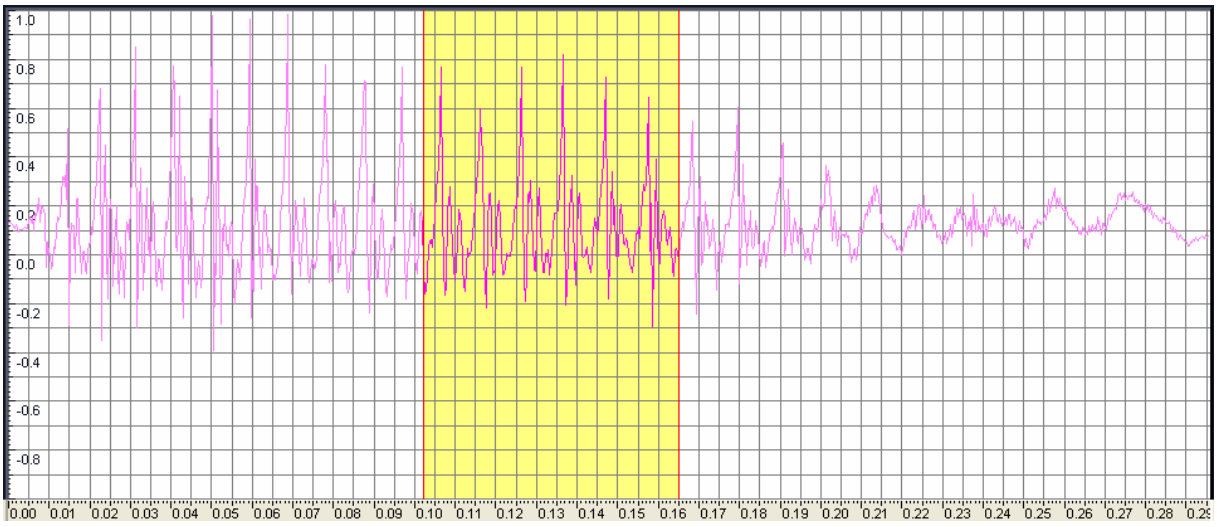
[9] “olabelované”
[10] v experimente som všetky samohlásky obaľoval fonémou „p“ a všetky spoluhlásky fonémou „a“
[11] pre jednoduchšie čítanie uvádzam grafémy



obr 8

zvuková vlna vzorky „pep“, „e“ je podsvietené (čas na vodorovnej osi je uvádzaný v sekundách)

Na [obr 8] vidno zvukovú vlnu nahrávky „pep”, prvých 20 milisekúnd je vlna fonémy „p”, plynule prechádza do vlny „e” , ktorá trvá zhruba 110 milisekúnd, za ňou nasleduje ticho (šum) a ďalšie “p”. Pri počúvaní tejto vzorky vieme pomerne presne[12] nájsť koniec „e” a menej presne ale postačujúco[12] začiatok „e”. Dá sa tiež orientovať podľa nápadnej zmeny amplitúdy vlny. Naopak pri určovaní hraníc vlny pre fonému „j“ sa dostávame do ťažkostí.



obr 9

zvuková vlna vzorky „aja“, „j“ je podsvietené (čas na vodorovnej osi je uvádzaný v sekundách)

[12] ANN v experimentoch pracuje s 5 alebo 10 ms časovým okienkom, čiže hranice fonémy môžu mať oddchýlku do 5 ms, ak ju majú aj do 10ms tak len jedno okienko bude trénované nesprávne

Na [obr 9] zvukovej vlny „aja“ nie je už žiaden nápadný rozdiel v amplitúde, a dokonca ani pri viacnásobnom počúvaní častí tejto vzorky nevieme povedať odkiaľ pokiaľ je fonéma „j“. Podsvietené hranice na obrázku treba chápať skôr ako miesto s vysokou pravdepodobnosťou výskytu esenciálnej vlny pre fonému „j“. Rozdelenie vzorky „aja“, ktoré je bližšie k realite (a k tomu ako počuje človek) je na fonému „a“, zhluk foném „-aja-“ a „a“. Nie všetky slová sa teda dajú rozdeliť na postupnosť foném.

Napovedá tomu aj pokus, ktorý som skúšal so vzorkami „aja“ a „ele“. Ak by sa dali tieto slová rozdeliť na fonémy „a“, „e“, „j“ a „l“, tak by malo byť možné pomocou nahradenia fonémy „l“ vo vzorke „ele“ fonémou „j“ zo vzorky „aja“ vytvoriť slovo „eje“. Podľa očakávaní sa ukázalo, že vznikne buď niečo podobné „eajae“ alebo „elje“[13]. Keďže vzorky musí vytvárať človek, bolo treba vziať do úvahy aj ľudské možnosti odlišovania foném. A tie boli pri označovaní vzoriek niekedy prekvapivé. Napríklad, ak počúvame nahrávku slova „šesť“ zhruba od konca „š“, tak celkom zreteľne počujeme „päsť“[13].

3.3 Fonémy zo vzoriek čísiel

Uvedené fakty z predchádzajúcej časti a výsledky experimentu so súborom foném 1 ma priviedli k tomu, že namiesto vzoriek niektorých foném treba vyskúšať robiť vzorky dvojíc alebo trojíc foném. Minimálne také fonémy, ktoré sa ukázali ako problematické - čiže krátke a s tendenciou plynule prechádzať do ďalšej fonémy. Taktiež bude treba nejako označovať aj ticho (šum) ktorý je vnútri slova medzi fonémami. Takéto ticho sa nachádza zvyčajne pred fonémami „p“, „c“, „t“, „ť“[14]. Ďalším podstatným zistením je, že výhodnejšie je robiť vzorky priamo zo slov, ktoré sa budú rozpoznávať. V nasledujúcom texte popíšem ako som postupoval pri vytváraní novej sady zvukových jednotiek (ktoré budú klasifikačnými triedami pre neurónovú sieť).

3.3.1 súbor rozpoznávaných zvukov 2

Fonémy som rozdelil do dvoch skupín. V prvej skupine sú fonémy, ktorým vieme pomerne jednoznačne určiť začiatok a koniec vo vzorkách slov. Sú to všetky samohlásky plus „s“ a „š“, pričom nerozlišujeme medzi dlhými a krátkymi samohláskami a medzi fonémou „a“ a „ä“ (dôvody sú uvedené v kapitole experimenty – súbor foném 1). Tieto fonémy som považoval za základné jednotky, ktoré klasifikovala aj neurónová sieť. Fonémy z druhej skupiny som už nepovažoval za základné jednotky, ale ich zlúčenia do dvojíc a trojíc s určitými pravidlami zlučovania.

[13] súbory na prehratie skladných zvukov a obrázky vĺn ostatných foném sa nachádzajú na priloženom CD
[14] napr. ticho - pauza pred „ť“ je niekedy aj šesťkrát dlhšia ako samotné „ť“

Prvým pravidlom je aby sa z nich z výslednej sady zvukov dali poskladať všetky rozpoznávané čísla (do milión). Druhým pravidlom je, že v sade zvukov je buď fonéma z prvej skupiny, alebo postupnosť foném z druhej skupiny, pričom táto postupnosť musí na začiatku a konci nadväzovať na nejakú fonému z prvej skupiny (výnimkou je len koniec a začiatok celého rozpoznávaného slova čísla[15]). Napríklad číslo jeden má transkripciu do tohto súboru zvukov: Je- E -eDe- E –eN[16]. Výsledný zoznam takto vytvorenej sady zvukov a transkripcie čísiel do nej je v dodatku 1 – *súbor foném 2*.

Pomocou tohto súboru základných zvukov (ktoré bude klasifikovať ANN) som obišiel problém určovania presných hraníc problematických foném z druhej skupiny. Napríklad hranice spomínaného „j“ budú označované v slove jeden až po moment pokiaľ „j“ bolo naozaj počuť, teda až za počiatočnú hranicu nasledujúcej fonémy „e“. Hranice jednotlivých rozpoznávaných zvukov sa teda budú prekrývať, čo by nemalo spôsobovať problémy pre rozpoznávanie celých slov čísiel. Očakávané správanie ANN na prekrývajúcich sa úsekoch je, že budú klasifikované ako obe fonémy[17].

Na takto vytváraných vzorkách foném som previedol druhý experiment rozpoznávania čísiel na jednom rečníkovi a ukázalo sa, že boli opäť nepostačujúce. Problém však už bol inde ako pri súbore foném 1. Pravdepodobne v tom, že niektoré fonémy sa vyskytovali v príliš veľa klasifikačných kategóriách (napr. „t“). ANN sa tak musela zbytočne sústrediť na rozdiely medzi „t“ zo vzorky eT a vzorky iaT (t a ť sa vyskytovali dohromady až v 12 kategóriách).

3.3.2 súbor rozpoznávaných zvukov 3

Zlúčením poznatkov z výsledkov z experimentov so súbormi zvukov 1 a 2 a zo skúseností z označovania ich vzoriek som dospel k názoru, že najvhodnejšie bude implementovať rozpoznávanie čísiel na uvedenom súbore zvukov v tejto časti[18]. Tento súbor zvukov sú v podstate fonémy zo súboru foném 1 s niekoľko malými rozdielmi: rozlišujeme medzi fonémou „v“, ktorá sa nachádza medzi dvoma „e“ (-eVe-) oproti „v“ pred ktorým je „d“ a nasleduje za ním samohláska (DVa-, DVe-), ďalej máme navyše zvukové jednotky – dvojice foném DN (-eDNa-) a DV (DVa-,DVe-). Zvukové jednotky T a TX sú fonémy „t“ a

---

[15] Keďže som implementoval rozpoznávanie celých slov čísiel a nie len foném (resp. zvukov skladajúcich sa z foném) tak som zaviedol aj podslová čísiel, z ktorých sa samotné čísla do milión skladajú (ich zoznam s transkripciou do príslušnej sady je v dodatku 1).

[16] Používam rovnakú konvenciu označovania zvukových jednotiek, ako v impelmentovanom programe: X označuje mäkčeň (tj napr. Ď označujem ako DX), pomlčka pred alebo za fonémou (označovanou malým písmenom) nazačuje nadväzovanie na túto fonému z prvej skupiny (čiže napr. „eDNa“ je zvuková jednotka v ktorej idú po sebe fonémy „d“ a „n“, pričom túto zvukovú jednotku predchádza fonéma „e“ a nasleduje ju fonéma „a“.

[17] teda aktivity výstupných neurónov budú vysoké pre obe fonémy

[18] Vznikol postupným zlučovaním zvukových jednotiek zo súboru 2, keďže najväčšie problémy spôsobovalo pravdepodobne veľa rôznych klasifikačných kategórií obsahujúcich rovnaké fonémy. Zlúčil som zvukové jednotky eT, sxTR, sxTi, sTo, CTo, iaT do spoločnej kategórie T' a sTX, eTX,aTX do spoločnej kategórie TX'. V týchto kategóriach sa už zbytočne vyskytovali aj fonémy "c" a "r", preto som "c" prehodil do kategórie C spolu s "c" zo zvukov iC a Ca, a založil špeciálnu kategóriu pre "r"

„ť“ spolu s tichom (šumom), ktoré vždy predchádza esenciálnu vlnu foném „t“ aj „ť“. Za zvukovú jednotku M (-eM,-eN) považujeme fonémy „m“ a „n“ predchádzané fonémou „e“.

V nasledujúcej tabuľke je zoznam zvukových jednotiek použitý v hlavnom experimente – rozpoznávania čísiel. (Keďže sú to väčšinou fonémy alebo im podobné jednotky zvukového systému jazyka, budem ich ďalej všetky nazývať fonémami. Dôležité je, že pre neurónovú sieť sú to klasifikačné kategórie a pre Viterbiho dekodér možné stavy.

	výskyt v číslach	označenie	obsahuje fonémy
1	je-	J	J
2	pe-	P	P
3	ďe-, -eďe-, -aď	DX	Ď
4	-ede-	D	D
5	-edna-	DN	D, N
6	dva-, dve-	DV	D, V
7	-eve-	V	V
8	-iri-, tri-, -štr	R	R
9	tri-, -štr, ti-, -t, to-	T	T
10	ťi-, -ť	TX	Ť
11	ná-, tná-	N	N
12	-em, -en	M	M,N
13	-a-	A	A
14	-e-	E	E
15	-i-	I	I
16	-ia-	IA	IA
17	-o-	O	O
18	-s-	S	S
19	-š-	SX	SX
20	-c-	C	C

Oproti súboru foném 1 je dôležitým rozdielom, že vzorky boli vytvárané zo skutočných slov čísiel. Preto sa však na nich už nedá použiť postup označovania vzoriek spriemerovaním hraníc a ich vylepšením označovacou ANN[19]. V práci je teda použité označovanie foném podobným spôsobom, ako korpus ručne označovaných nahrávok hovoreného slova – TIMIT, ktorý sa bežne používa pri výskumoch o rozpoznávaní angličtiny. Zoznam nahrávaných slov je v dodatku 2.

[19] Je totiž potrebné rozďeľovať nahrávané slová na menšie úseky, v ktorých by sa vyskytovali maximálne len dve fonémy, keďže predpokladám že označovacia ANN môže fungovať len na rozďeľovaní zvokov do dvoch kategórií. Tým pádom som sa rozhdol rozďeľovať slová rovno na zvukové jednotky, ktoré budú na výstupe ANN, čo je už vlastne klasické označovanie.

4 Neurónová sieť

V tejto kapitole opíšem parametre a architektúru klasifikačnej neurónovej siete použitej na rozpoznávanie foném. Taktiež techniky ktoré som skúšal aplikovať na vylepšenie úspešnosti klasifikátora. Nakoniec uvediem aké hodnoty som používal pri hľadaní optimálnych parametrov v experimentoch.

4.1 Vstup do neurónovej siete

Na vstupnú vrstvu neurónov budú prichádzať pozorovania z jednotlivých časových okienok analyzovanej zvukovej vlny. Ako som už uviedol v časti o extrakcii pozorovaní, na výber som mal Mel-frekvenčné kepstrálne koeficienty (MFCC)[20], LPC koeficienty (LPC), kepstrálne LPC koeficienty (LPCEPSTRUM) a energiu signálu (E). Pri výbere koeficientov som sa inšpiroval hlavne prácou [TSR], kde sa využíval dvanásť-rozmerný vektor MFCC koeficientov spolu s normalizovanou energiou signálu. Navyše pre všetky komponenty tohto trinásť-rozmerného vektora ich prvé a druhé derivácie v čase[21]. Vo väčšine trénovaní ANN som využil tento 39 rozmerný vektor príznakov na jedno okienko.

Časové rozhranie okienka[22] som volil päť alebo desať milisekúnd, pričom pozorovania pre tieto oblasti sa vytvárali z 12,5 milisekundových oblastí (pre 5 milisekudové okienko) alebo 25 milisekundových oblastí (pre 10 milisekudové okienko). Napriek tomu, že v prácach [MBA] a [ASR] sa používali 10 milisekundové okienka, rozhodol som sa väčšinou využívať 5 milisekundové, pretože niektoré (rýchlejšie vyslovené) vzorky krátkych foném mali dĺžku len 20 milisekúnd.

Keďže aktivity neurónov v použitej ANN sú v intervale [-1;1], tak hodnoty na vstupe, ktoré by boli výrazne mimo tohto intervalu by znižovali úspešnosť trénovania. Preto sú hodnoty pozorovaní normalizované do intervalu [-1,1]. Dôvody sú uvedené aj v dokumentácii použitého toolkitu na vytváranie neurónových sietí NICO [NIK]. Lineárna transformácia, ktorá mapuje maximálnu hodnotu zo všetkých pozorovaní do +1 a minimálnu do -1 tiež nie je celkom vhodná. Vrstva vstupných neurónov by bola vo veľkej miere citlivá len na vyššie hodnoty. Preto som používal normalizáciu založenú na priemere a štandardnej odchýlke vstupných hodnôt pozorovaní. V tejto sa mapuje do +1 hodnota: priemer + **d** * štandardná

[20] V zátvorkách sú skratky používané v HTK
[21] V konfigurácii HTK sa označuje takýto vektor pozorovaní ako TARGETKIND = MFCC_E_D_A
[22] V názvosloví HTK - TARGETRATE

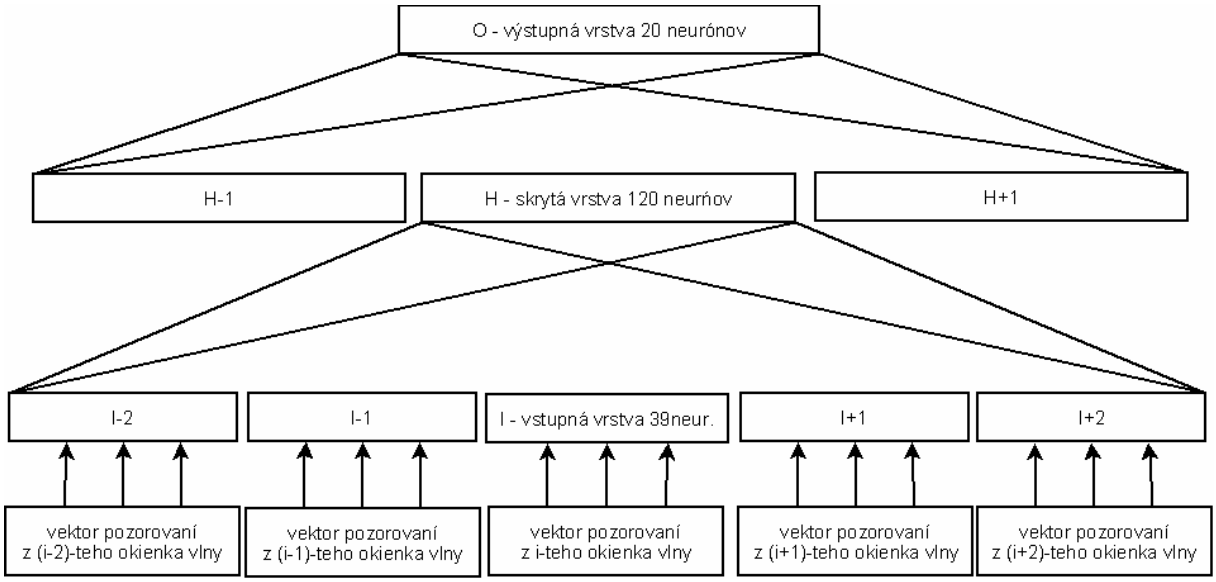
odchýlka (a priemer - **d** * štandardná odchýlka do -1)[23]. Parameter **d** je voliteľný, a tak bolo treba robiť aj trénovania, na základe ktorých som hľadal jeho optimálnu hodnotu. Príliš vysoká hodnota sa už začína podobať na priamu lineárnu transformáciu, príliš nízka hodnota je zase citlivá len na malé hodnoty (menšie ako -1).

4.2 Architektúra klasifikačnej neurónovej siete

V implementácii som používal troj-vrstvovú architektúru klasifikačnej ANN, tak ako v prácach [MBA] alebo [ASR]. Na vstupnú vrstvu prichádzajú hodnoty pozorovaní z jednotlivých okienok rozpoznávanej vlny. Tá je plne prepojená s vrstvou skrytých neurónov, ktorá je zase plne prepojená s výstupnou vrstvou. Vo výstupnej vrstve je toľko neurónov, koľko máme klasifikačných tried, čiže počet foném. Použil som lokalistické kódovanie, čiže očakávaným výstupom pre i-tu fonému na vstupe je vektor, ktorý ma na i-tom mieste +1 a na všetkých ostatných -1. Najviac trénovaní ANN som skúšal s počtami neurónov v skrytej vrstve v rozmedzí 50 – 200. Mimo tohto rozsahu som skúsil len niekoľko simulácií, v ktorých sa potvrdil predpoklad, že úspešnosť klasifikátora sa zhorší.

Rozpoznávaná vlna zvuku je podelená na 5 alebo 10 milisekundové úseky okienok a fonémy majú dĺžku v intervale zhruba 20 až 300 milisekúnd. Na vstup neurónovej siete teda môže niekedy prichádzať až vyše 50 vektorov pozorovaní tej istej fonémy za sebou. Aby som v architektúre ANN zahrnul aj tento časový aspekt reči využil som (podobne ako napr. v práci [MBA]) techniku neurónových sietí s oknom do minulosti (*time delay neural network* TDNN). Teda skrytá vrstva bude pri klasifikovaní (a teda aj trénovaní) daného okienka aktivovaná nie len hodnotami zo vstupnej vrstvy, ktorá bola aktivovaná vektormi príznakov daného okienka, ale aj hodnotami ktoré vznikajú na vstupnej vrstve, ak sú na vstupe susedné okienka. TDNN má takto zrealizované "okno do minulosti a budúcnosti". Rovnaká techinka časového kontextu môže byť aplikovaná aj o úroveň vyššie – medzi skrytou a výstupnou vsrtvou neurónov. Ďalšími parametrami architektúry ANN sú teda veľkosti kontextu (okien do minulosti a budúcnosti) skrytej a výstupnej vrstvy. Na obrázku [obr 10] je príklad[24] architektúry použitej TDNN s veľkosťou kontextu 5 pre skrytú vrstvu (tj skrytá vrstva je aktivovaná vstupnými vrstvami z časov -2 až +2) a 3 pre výstupnú vrstvu.

[23] Využíval som na to modul z NICO toolkitu NormStream
[24] Toto bola zároveň aj najúspšnejšia architektúra

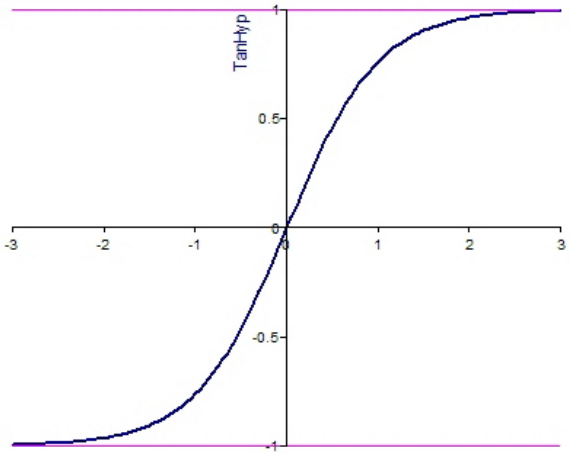


obr 10

V takejto ANN je pochopiteľne omnoho viacej prepojení neurónov. Keby sme nepoužívali kontextové prepojenia, tak máme počet prepojení I * H + H * O, kde I je počet neurónov vo vstupnej, H skrytej a O výstupnej vrstve. V ANN z obrázku [obr 10] je to 5*I*H + 3*H*O prepojení, čiže pre používané parametre I=39, H=120, O=20 až 30 600 prepojení oproti 4820 prepojeniam v ANN bez časových okien. Príliš vysoký počet prepojení už úspešnosť klasifikátora zhoršuje a navyše výpočet jednej iterácie trénovania sa takto zšesťnásobila. Preto som skúšal kontexty v obmedzenom rozmedzí -6 až 6 pre skrytú a -2,2 pre výstupnú vrstvu.

Za aktivačnú funkciu neurńov som zvolil hyperbolický tangent [obr 11] s predpisom:

$$TanHyp(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad .$$



obr 11

Podľa dokumentácie [NIK] je ju výhodnejšie používať než viacej známu bipolárnu sigmoidu s prepdisom $s(x) = \frac{2}{1+e^{-\lambda * x}}$, kde λ je strmosť sigmoidy. Navyše sú ANN vytvorené v NIKO toolkite optimalizované pre aktivačnú funkciu tanhyp.

Vo väčšine trénovaní som používal štadardnú chybovú funkciu – sumu štvorcov chýb na všetkých výstupných neurónoch: $E = \frac{1}{2}\sum_{i=1}^{K}(d_i - o_i)^2$, kde K je počet klasifkovaných tried, **d** – vektor očakávaných výstupov a **o** vektor aktivít výstupnej vrstvy. Podľa odporúčaní (napr. z [NIK]) som vyskúšal ako chybovú funkciu vzájomnú entropiu (*cross entropy*) s predpisom: $E = -\sum_{i=1}^{K} d_i * \ln(o_i)$. Mala by byť alternatívou v kalsifikačných problémoch, kde aktivity výstupnej vrstvy neurónov sú chápané ako pravdepodbnosti jednotlivých kategórií.

4.3 Trénovanie neurónovej siete

Parametre samotného trénovania neurónovej siete, ktoré najviac ovplyvňovali úspešnosť klasifkovania boli rýchlosť učenia[25] (*gain*) s ňou silne súvisaci počet iterácií trénovania a momentový člen[26] (*momentum*). Okrem týchto som sledoval ako ovplyvňujú úspešnosť klasifkátora aj nasledujúce parametre (resp. techniky) vylepšenia trénovania.

Tlmenie váh (*weight decay*) – parameter ktorý hovorí o sile zmenšovania váh neurónu po každej iterácii: v ← (1 – w) * v, kde **w** je parameter weight decay, a **v** váha. Takto sa urýchľuje proces eliminácie prepojení neurónov aj neurónov samotných (*pruning*). V použitej ANN bolo implementované toto rušenie spojov (s nízkymi váhami) aj neurónov (s malými príspevkami).

Pomocou validačnej množiny (ktorou bolo zhruba 10% vzoriek z každej fonémy) som aplikoval metódu skorého zastavenia (*early stopping*)[27] a redukovanie rýchlosti učenia počas trénovania. Čiže keď sa chyba na validačnej množine medzi dvoma epochami nezmenší, tak sa rýchlosť učenia zníži - vynásobí sa redukčným koeficientom (menším ako 1), to však najviac určitý počet krát. Tento počet je druhým parametrom techniky dynamického znižovania rýchlosti učenia.

[25] Rýchlosť učenia hovorí o sile aplikovania (násobku) učiaceho pravidla neurónu, detialy v [UNS]
[26] Momentum hovorí o sile aplikovania (násobku) učiaceho pravidla neuurónu z predošlej iterácie, detaily v [UNS]
[27] Ak začne chyba na validačnej množine rásť, trénovanie sa zastaví, detaily v [UNS]

4.4 Sledovanie úspešnosti trénovania

Aby som našiel parametre, čo najpodbnejšie optimálnym, sledoval som viacero ukazovateľov úspešnosti ANN. Jednak veľkosť chybovej funkcie na trénovacej aj validačnej množine a ich vývoj v čase. Ďaľším a zrozumiteľnejším ukazovateľom je úspešnosť samotného klasifikovania (*frames correct*). Teda koľko percent z okienok vzoriek foném bolo označených ako fonéma, z ktorej dané okienko pochádza. Pričom za klasifikovanie i-tej fonémy považujeme to, keď i-ty neurón vo výstupnej vrstve mal najväšiu aktivitu (z neurónov vo výstupnej vrstve). Podobne môžeme vyhdnotiť aj úspešnosť klasifikovania druhej úrovne, kde. za správne oklasifikovanie vzorky (úseku vlny i-tej fonémy) budeme považovať, ak neurón z výstupnej vrstvy s maximálnou alebo druhou maximálnou aktivitou bol i-tym neurónom výstupnej vrstvy. Na sledovanie úspešností jednotltivých foném a podbností foném medzi sebou je užitočná klasifikačná tabuľka (*confusion table*). V nej je v r-tom riadku a s-tom stĺpci uvedené na koľko percent bola r-tá fonéma klasifikovaná ako s-tá fonéma. Vďaka tejto tabuľke som sa ľahšie rozhodoval v experimentoch s prvými súbormi jednotiek zvukov, ktoré klasifikačné kategórie zlúčim, resp. rozdelím[28]. Taktiež ovplyvňovala moje rozhodnutia o tom, aké slová vložím do slovníka pre gramatiku (viď kapitola – Rozpoznávanie slov čísiel). Obe úspešnosti klasifikovania aj klasifkačnú tabuľku som sledoval pre trénovaciu aj validačnú množinu.

Pre trénovania, ktoré vykazovali najlepšie úspešnosti podľa už uvdených hodnôt, som testoval aj úspešnosť v rozpoznávaní celých slov čísiel. Predpokladal som, že nemusí vždy platiť priama úmera (najmä pri malých rozdieloch úspešnsti ANN) medzi úspšnosťou klasifikovania ANN a rozpoznávaním celých slov čísiel.

[28] Napríklad rozhodnutie zlúčiť fonémy „m“ a „n“ pred ktorými je „e“ do spoločnej kategórie

5 Rozpoznávanie slov čísiel

V tejto kapitole opíšem poslednú fázu rozpoznávača čísiel. Akým spôsobom sa určí konkrétne slovo zo slovníka na základe výstupu z neurónovej siete. Ktoré slová budú v slovníku a aké vety sa z nich budú dať vytvoriť. A na záver algoritmus rozpoznávača čísiel do milión.

5.1 Rozpoznávané slová

Úlohou celého systému je rozpoznávať slová[29], z ktorých sa dajú poskladať prirodzené čísla do milión (v slovenčine). Podobne ako v práci [MBA], kde sa rozpoznávali slová, z ktorých sa dajú poskladať (anglické) čísla do 100. Slová rozpoznávané v tejto práci sú uvedené v nasledujúcej tabuľke:

jeden	jede	
dva	dve	
tri		
štyri		
päť	pät	pä
šesť	šes	
sedem		
osem		
deväť	devät	devä
desať		
desiat		
štr		
násť		
cať		
sto	cto	
tisíc		

Na rozdiel od práce [MBA] skladám čísla medzi desať a dvadsať napr. ako dva + násť. Takýto spôsob pokladám za výhodnejší, čo sa týka úspešnosti na celých vetách z tohto slovníka, pretože slovo násť sa nevyskytuje zbytočne vo viacerých slovách. Rozpoznávač nebude rozlišovať medzi slovami, ktoré sú uvedené v jednom riadku tabuľky. Predpokladám že pre potreby rozpoznávania čísiel zlúčených z týchto slov to nemá veľký význam. Jedinú informáciu, ktorú by nám poskytol rozdiel napr. medzi „deväť“ a „devä“, je že za „deväť“ musí nasledovať „násť“ a za „devä“ musí byť „desiat“. Pre rozpoznávač je však jednoduchšie odlíšiť „násť“ od „desiat“ ako „devä“ od „deväť“. Navyše prechody medzi slovami z tohto

[29] Tieto slová nemusia byť spisovnými slovami slovenského jazyka

slovníka sú pri vyslovovaní čísiel plynulé a nedá sa presne určiť kedy začína a končí slovo. A keď nebudem rozlišovať medzi „deväť“ a „devä“, tak ponechávam určitú voľnosť pri klasifikovaní napr. čísla „deväťťisíc“, ktoré ľudia vyslovujú s jedným alebo dvoma „ť“.

V slovníku je teda 16 kategórií slov, pričom rozpoznávač nerozlišuje medzi slovami v jednej kategórii. Inak povedané - kategóriu slov môžeme chápať ako jedno slovo, ktoré má viacero možností vyslovenia.

5.2 preloženie výstupu ANN do slova

Výstupom neurónovej siete sú pravdepodobnosti rozpoznania foném pre jednotlivé časové okienka. Ako som popísal v časti 2.3.3 z tejto matice pravdepodobností skladá Viterbiho algoritmus optimálnu sekvenciu foném do slova. Rieši tak problém optimálnej postupnosti stavov Q = { q_1 q_2 … q_t }pre danú postupnosť pozorovaní O = {O_1 O_2 … O_t} podľa induktívnej definície δ_t:

$$\delta_1(j) = \pi_i * b_i(O_1)$$

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] * b_j(O_{t+1})$$

kde π je vektor pravdepodbností počiatočného stavu, a_{ij} je pravdepodobnosť prechodu zo stavu i do stavu j, $b_j(O(t))$ je pravdepodobnosť stavu j v čase t. δ_t teda maximálna pravdepodobnosť stavov po čas t. Pri rozpoznávaní sú stavmi fonémy a teda hodnoty $b_j(O(t))$ sú výstupom ANN. Vektor π a maticu **A**=[a_{ij}] je treba vypočítať vopred[30]. Jej definícia je: $a_{ij} = \frac{\sum_{w \in D} T(i,j,w)}{\sum_{w \in D} L(w)}$ kde T(i,j,w) je počet prechodov z i-tej fonémy do j-tej fonémy v slove **w**, L(w) je počet foném v slove **w**, D je množina slov slovníka, pričom slovo **w** je postupnosť foném v čase v slove w. Zrejme si treba zvoliť nejakú časovú jednotku, ktorá bude menšia ako veľkosť používaného okienka pozorovaní. Potom získava funkcia L(w) význam dĺžky vzorky slova w a funkcia T(i,j,w) je vlastne súčtom dĺžok vzorky i-tej fonémy v slove w pre i = j, a počet prechodov z i-tej do j-tej fonémy v transkripcii[31] slova w pre i ≠ j. Konkrétne hodnoty som získal z prepisov jednotlivých slov zo slovníka do stavov foném v čase. Napr. slovo sedem (S_E_DX_E_M) ma takýto prepis:

S 0 150
E 146 260

[30] Pri rozpoznávaní reči HMM sa podbná matica získa trénovaním HMM.
[31] Foneitckom prepise

DX 235 337
E 315 530
M 503 683

Je to vlastne označenie vzoriek použité aj pri trénovaní ANN a hovorí o tom, koľko časových jednotiek[32] trvajú jednotlivé fonémy a ako po sebe nasledujú. Veľkosť časovej jednotky ovplyvňuje pomernú veľkosť pravdepodobnosti prechodov medzi rôznymi fonémami oproti pravdepodobnosti prechodu na rovnakú fonému (t.j. zotrvaniu na rovnakej foném). V testoch[33], sa potvrdil predpoklad, že optimálna časová jednotka je dĺžka časového okienka. Vektor π hovorí o tom, ktoré fonémy môžu byť na začiatku slova. Má definíciu: $\pi_i = \frac{\sum_{w \in D} S(i,w)}{\sum_{w \in D} L(w)}$, kde S(i,w) je pravdepodobnosť že prvým stavom v slove w je i-ta fonéma.

V praxi stačí do π_i priradiť nenulovú hodnotu práve vtedy, keď existuje slovo zo slovníka začínajúce i-tou fonémou, čo ľahko zistíme z transkripcií jednotlivých slov.

Viterbiho algoritmus s takto definovanou maticou prechodov však málokedy vytvorí postupnosť foném, ktorá by bola slovom zo slovníka. Keď odignorujeme vo výstupe z Viterbiho dekodéra prechody medzi rovnakými fonémami[34], dostaneme napr. pre testovaciu vzorku „sedem“ výstup SX E S TX I E DX E M. Je to preto, že matica prechodov bola vytvorená zo všetkých slov slovníka. Ak teda ANN neklasifikuje jednotlivé okienka vzorky dostatočne presne, tak optimálnou postupnosťou foném s danou maticou prechodov, môže byť uvedená „nechcená“ postupnosť foném.

Vhodným riešením problému rozpoznávania slov zo slovníka je teda nasledovný postup. Pre každé slovo zo slovníka vygenerujeme osobitnú maticu prechodov[35] aj vektor pravdepodobností počiatočného stavu. Vypočítame pravdepodobnosť optimálnej postupnosti foném pre všetky slová s použitím ich matice prechodov a maximalizáciou týchto pravdepodobností určíme aké slovo sme rozpoznali. Toto je hlavná myšlienka implementácie poslednej fázy rozpoznávania celých slov. Navyše som ju vylepšil ďalšími technikami, ktoré opíšem v nasledujúcom texte. Celková časová zložitosť je počet slov v slovníku krát zložitosť Viterbiho algoritmu, čiže $O(W * N^2 * T)$, kde W je počet slov v slovníku, N počet foném, T je okienková dĺžka rozpoznávaného slova.

[32] Tu sú uvedené milisekundy
[33] V programe sa dá voliť tento pomer pomocou konštánt TRANSITION_BONUS a STAY_PENALTY
[34] Takúto skrátenú podobu postupnosti foném budem ďalej nazývať slovotvarom tejto postupnosti
[35] Matica prechodov pre jedno slovo má rovnakú definíciu, a teda aj výpočet ako matica prechodov pre všetky slová zo slovníka, jediným rozdielom je množina slov D, kde sa nachádza len dané slovo

5.3 Vylepšenia poslednej fázy rozpoznávania slov

V tejto časti opíšem rôzne techniky, ktorými som vylepšoval úspešnosť rozpoznávania slov na základe experimentálnych zistení z testov rozpoznávania slov.

5.3.1 Topológia slov

Ako som pozoroval v experimentoch, ani vlastná matica prechodov pre každé slovo nezaručuje, že optimálna postupnosť foném (v slovotvare)vypočítaná Viterbiho dekodérom bude zodpovedať slovu zo slovníka. Stále totiž nie je zaručené, že optimálna postupnosť foném nebude napr. pre vzorku „sedem“ S_E_DX_E_DX_E M, ak ANN nevedela uprostred vzorky dobre klasifikovať ď (DX), alebo S_E_DX_E, ak mala ANN problémy s klasifikovaním vysloveného „m“. Inšpiráciou z prednášok [HMM] som došiel k predpokladu, že by bolo vhodné do Viterbiho algoritmu nejako zahrnúť aj topológiu slova. Teda určit aké sú apriórne pravdepodobnosti výskytu foném v časti slova. Tým som chcel zamedziť aby optimálne postupnosti z Viterbiho algoritmu, ktoré sa nepodobajú slovu zo slovníka neboli optimálnymi a „nechali tak priestor” pre ostatných kandidátov na optimálnu postupnosť.

Topológiu foném v slove som modeloval pomocou Normálneho[36] rozdelenia pravdepodobnosti: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-1/2\left(\frac{x-\mu}{\sigma}\right)^2}$ pričom strednú hodnotu μ som určil ako stred intervalu výskytu danej fonémy vo vzorke a odmocninu z disperzie σ som zisťoval experimentálne, najlepšie výsledky boli so σ = 0,4. Samotné včlenenie topológie do Viterbiho algoritmu som implementoval pomocou matice označovanej v programe ako wordTopo[37], kde člen $wordTopo_{it}$ je rovný pravdepodobnosti i-tej fonémy v t-tom okienku daného slova[38]. Keďže počet okienok klasifikovaného slova môže byť rôzny, tak časový rozmer matice wordTopo bol premenlivý a rovnal sa vždy počtu okienok klasifikovaného slova. Viterbiho algoritmus s topológiou už teda nepoužíva ako maticu pozorovaní výstup z ANN, ale členy matice pozorovaní B'=[b'_{it}] sú definované ako $b'_{it} = b_{it} + wordTopo_{it}^{TP}$ kde b_{it} je člen matice B - pravdepodobností foném v čase (výstup ANN) a člen **TP** je koeficient[39], ktorý hovorí o sile včlenenia topológie do Viterbiho algoritmu, ktorý som zisťoval experimentálne. Potvrdili sa intuitívne predpoklady. Jeho optimálna veľkosť závisela od úspešnosti ANN, čím lepšia úspešnosť ANN, tým menší koeficient **TP**. V trénovaniach na jednom rečníkovi s málo vzorkami bol optimálne TP = 0,5 ale v trénovaniach s viac rečníkmi s osem-násobným

[36] Gaussovského
[37] Je to rovnaký princíp ako včlenenie vektora π do Viterbiho dekodéra, ibaže po celej dĺžke slova
[38] Pochopiteľne pre každé slovo bola špeciálna matica.
[39] V kóde prgramu sa volá TopoPower

počtom vzoriek bolo optimálne TP blízko 0, alebo rovné nule čo znemená, že topológia slova sa nevyužívala.

5.3.2 Skóre rozpoznaného slova

V časti 5.2 som uviedol, že víťazné (t.j. rozpoznané) slovo, je to, ktoré má maximálnu hodnotu svojej[40] optimálnej postupnosti foném. Experimentálne sa ukázalo, že nie je vhodné určovať víťazné rozpoznané slovo len na základe hodnôt $DS_w = \max_i \delta_{iT}$ kde T je index posledného okienka rozpoznávaného slova, w je index slova – kandidáta na víťazné rozpoznané slovo. Pre každé slovo máme k dispozícii okrem hodnoty DS_w[41] aj ďalšie hodnoty, ktoré môžu byť pomerne zaradené do celkového skóre pre slovo.

Ako sa ukázalo z experimentov, najpodstatnejším komponentom výsledného skóre je $BS_w = \prod_{t=1}^{T} B_{st}$ kde s = $fonemy_t$, pričom fonemy je pole indexov víťazných foném, ako ich určil Viterbiho dekodér a **B** je matica pravdepodobností foném v čase (výstup ANN). Hodnota BS[42] pre slovo w, teda vraví o tom, ako sa to čo sme pozorovali neurónovou sieťou zhoduje s optimálnou postupnosťou foném podľa Viterbiho dekodéra.

Keďže ani Viterbiho algoritmus s topológiou nezaručuje, že optimálna postupnosť foném (v slovotvare) pre dané kandidátske slovo s jeho maticou prechodov nebude slovom zo slovníka. Preto ako ďalší komponent celkového skóre slova – kandidáta by mala byť vzdialenosť medzi postupnosťou foném kandidátskeho slova a postupnosťou foném (v slovotvare) ktorú považoval Viterbiho algoritmus za optimálnu. Ako túto vzdialenosť som definoval LS[43] = Levenshtein-ovej vzdialenosti[44]. Čiže ak budeme rozpoznávať slovo „deväť“, mal by mať vyššie skóre kandidát DX_E_V_E_TX než kandidát DX_E_V_E.

Posledným komponentom celkového skóre je FTS[45] rozdiel priemernej časovej dĺžky kandidátskeho slova (získanej z trénovacích vzoriek) a rozpoznávaného slova.

Všetky uvedené komponenty DS, BS, FTS, LS sú mapované do pravdepodobnostného intervalu <0,1> pomocou Normálnych rozdelení pravdepodobností, ktorých parametre som určoval experimentálne. Celkové skóre pre kandidátske slovo vypočítavam ako vážený priemer týchto komponentov[46], pričom ich zastúpenia som tiež zisťoval experimentálne.

---

[40] Vzhľadom na svoju maticu prechodov
[41] V kóde programu sa volá D_score
[42] V kóde programu sa volá B_score
[43] V kóde programu sa volá L_score
[44] Je to minimálny počet operácií potrebný na transformovanie jednej postupnosti do druhej, pričom povolenými operáciami sú zmazanie, vloženie a nahradenie.
[45] V kóde programu sa volá FT_score
[46] V kóde programu sa skóre počíta vo funkcii eval_candidate

5.4 Rozpoznávanie čísiel do milión

Keďže z rozpoznávaných slov uvedených v časti 5.1 sa dajú skladať prirodzené čísla do milión, rozhodol som sa ako sekundárny cieľ zistiť aj úspešnosti rozpoznávania čísiel do milión. Čísla vyslovujeme plynule a medzi slovami, z ktorých sa čísla skladajú, nie sú prestávky ticha[47], preto sa nedá riešiť problém rozpoznávania čísiel ako izolovaných slov (zo slovníka v časti 5.1) vo vete. V tejto časti predvediem prácu rozpoznávača pomocou dvojice konečných automatov so zásobníkom a systému na rozpoznávanie slov, ktorý som popisoval v doterajších kapitolách.

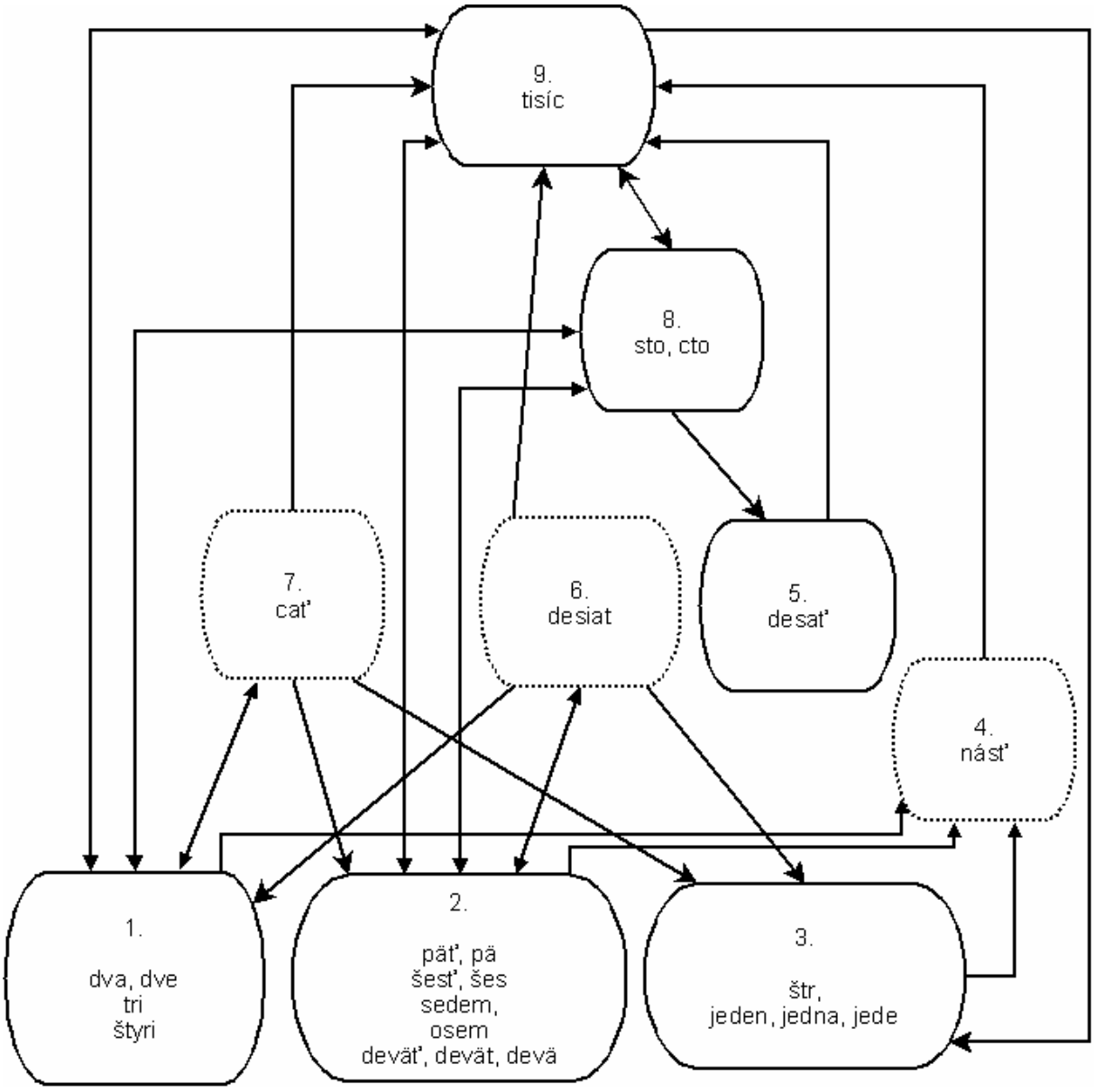
Čísla do milión som rozpoznával pomocou nasledovného „backtrack" algoritmu[48]:

vytvor výstup ANN z celej vzorky slova pre Viterbiho alg.
1. incializuj nasledujúcu fázu (prehľadávanie do hĺbky)
ta **,** (aktuálny potenciálny začiatok slova),
deep (aktuálna hĺbka prehľadávania),
scope[deep], (aktuálna šírka prehľadávania)
ak *ta* je koniec rozp. vlny zaznemaj do poľa výsledkov rozpoznané číslo
(Pomocou poľa kandidátskych slov vypočítaj číslo a skóre tohto čísla, pole výsledkov udržuj usporiadané od najlepšieho skóre pre číslo po najhoršie)
ak prehľadávanie do hĺbky bolo ukončené, vypíš výsledky a skonči

2. Indentifikuj okienka na vzorke, ktoré môžu byť potenciálnymi konca slova[49]
(To pomocou Viterbiho algoritmu na celej rozpoznavanej vlne, s použitím matice prechodov pre všetky slová v slovníku. Potenciálnymi koncami slova budú okienka kde sa prechádza na rôznu fonému)
3. Pre všetky potenciálne konce slova *t*
4. Pre všetky slová *w* zo slovníka
Ak stav slova *w* vyhovuje automatu 1
A dekadický stav stav slova *w* vyhovuje automatu 2
Vypočítaj skóre slova *w* na vzorke od okienka *ta* po okienko *t*
(To pomocou Viterbiho algoritmu s maticou prechodov pre slovo w a funkcie pre skóre slova uvedenej v časti 5.3.2)
Zaktualizuj pole kandidátskych slov
(Ak si našiel lepšie skóre pre nejakú trojicu: ciferná hodnota slova,stav slova,potenciálny koniec slova, tak si zaznamenaj potrebné informácie o tomto slove do poľa kandidátov pre aktuálnu hĺbku prehľadávania *deep* pričom udržuj toto pole usporiadané od najlepšie skóre po najhoršie)
5. choď do kroku 1.

[47] Aj keď by sa mohlo zdať, že napríklad stojeden vyslovujeme oddelene s krátkou pauzou medzi sto a jeden, nie je tomu tak, vlna zvuku prechádza plynule z fonémy „o" na „j", tiché prestávky medzi fonémami nájdeme skôr vnútri slova ako medzi slovami, napr. v slove päť pred fonémou „ť"
[48] Tento alogirtmus je nepodloženým neakademickým pokusom, v programe je to funkcia recognize_word
[49] Slovom budem nazývať slovo zo slovníku, vetou rozpoznané celé číslo do milión

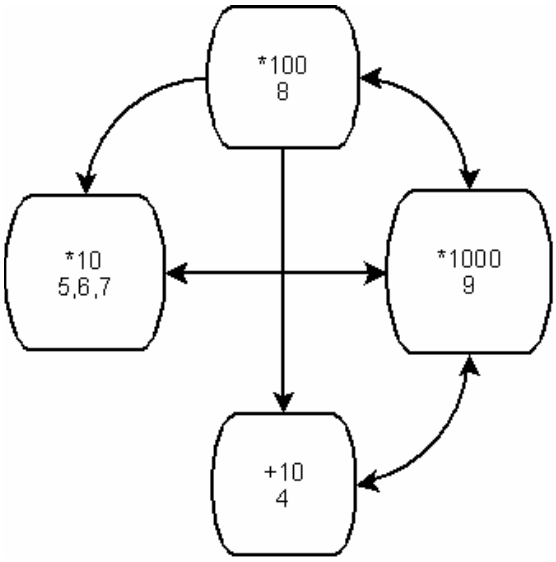
Pod slovným spojením v pseudokóde „stav slova w vyhovuje automatu“, mám namysli, že existuje prechod v automate z aktuálneho stavu cez slovo w do nasledujúceho stavu. Nasledujúci obrázok [obr 12] znázorňuje prechodovú funkciu Automatu 1.



obr 12

Pre väčšiu prehľadnosť sú symboly prechodovej funkcie zakreslené vnútri stavu. Čiže symboly automatu (v našom prípade rozpoznávané slová), ktoré sú uvedené vnútri stavu majú byť nad každou šípkou, ktorá do daného stavu vchádza. Všetky stavy môžu byť počiatočné aj finálne, okrem orámovaných prerušovanou čiarou.

Vďaka automatu 2 [obr 13] vieme určiť aj aké viacciferné číslo sme rozpoznali. Jeho symolmi sú stavy z automatu 1. Okrem týchto dvoch automatov ešte kontrolujem, či sme v stave *100 boli len raz.



obr 13

6 Experimenty

6.1 Súbory foném 1 a 2

Prvé testy som previedol na súbore foném 1, ktorých zoznam aj s transkripciou je v dodatku 1. Vzorky boli vytvárané nevhodným spôsobom (ako útržky neexistujúcich slov), kvôli čomu sa znižovala schopnosť neurónovej siete zovšeobecňovať. Tieto testy aspoň upozornili na podobnosť niektorých foném ako vidno v nasledujúcej klasifikačnej tabuľke [tab 1].

	a	ax	á	ä	c	d	ď	e	i	í	ia	j	l	m	n	o	ó	p	p	r	s	š	t	ť	u	v
a	73	1	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	5	0	0	0	0	0	0	1
ax	1	85	3	1	0	1	0	0	0	0	1	1	1	0	0	0	0	0	3	1	0	0	0	0	0	0
á	1	1	97	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ä	0	0	0	85	0	0	2	10	0	0	1	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0
c	0	0	0	0	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	1	0	0	0
d	0	9	0	0	1	75	1	0	0	0	0	0	0	0	0	0	0	1	6	0	0	0	7	1	0	0
ď	0	1	0	1	0	3	89	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	1	0	0	0
e	0	0	0	48	0	0	0	45	0	0	2	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	76	21	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
í	0	0	0	0	0	0	0	0	24	75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ia	0	0	0	2	0	0	0	2	0	0	90	1	0	0	0	0	0	1	3	0	0	0	0	0	0	0
j	0	0	0	1	0	0	0	0	0	0	3	96	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	0	2	0	0	0	0	0	0	0	0	0	0	97	0	0	1	0	0	0	0	0	0	0	0	0	0
m	0	1	0	0	0	0	0	0	0	0	0	0	0	1	98	0	0	0	0	0	0	0	0	0	0	0
n	0	3	0	0	0	0	0	0	0	0	0	0	0	0	97	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	81	17	0	1	0	0	0	0	0	2	0
ó	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	73	0	1	0	0	0	0	0	0	0
p	4	23	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	69	1	0	0	0	0	0	1	0
px	1	1	1	1	0	0	0	1	2	1	1	0	0	0	0	1	1	0	88	0	0	0	0	0	1	0
r	3	12	7	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	5	72	0	0	0	0	0	0
s	0	2	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	92	0	0	0	0	0
š	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98	0	0	0	0
t	0	6	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	83	5	0	0
ť	0	0	0	0	1	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	76	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4	0	0	0	0	0	95	0
v	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	99

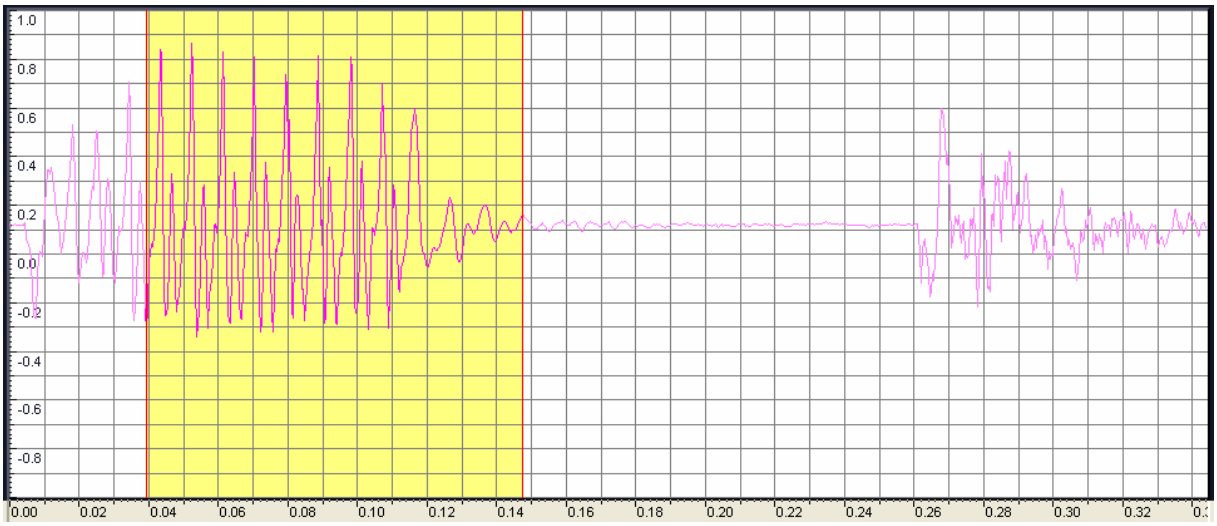
tab 1

Klasifikačná tabuľka trénovania na 1 rečníkovi, 88 skrytých neurónov, 1000 iterácií, rýchlosť učenia 1,00E-5, momentum 0,1

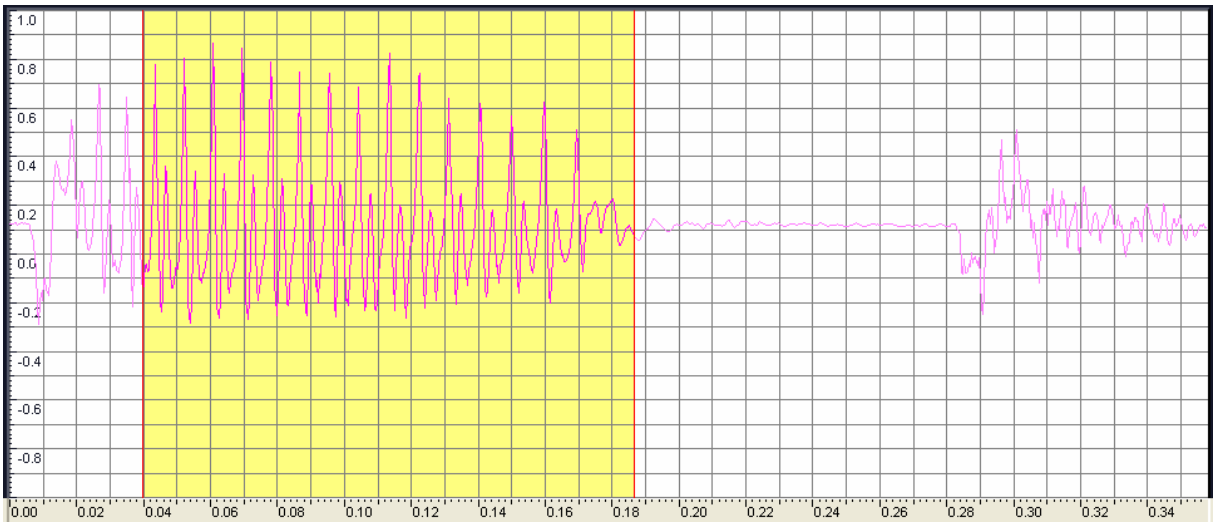
Kategórie ax a px, netreba brať do úvahy, boli to vlny obaľovacích foném „a“ a „p“. Rozlišoval som medzi fonémou "p" ktorá sa nachádza vo vzorkách typu p+samohláska+p, (napr. "pap", "pep") a fonémou "p", ktorú sa sieť učila zo vzorky "apa" analogicky som rozlišoval "a" zo vzoriek a+spoluhláska+a od "a" zo vzorky "pap".

Jasne je z tejto tabuľky vidieť, že ANN mala problémy rozlišovať medzi „a“ a „á“, „i“ a „í“, „o“ a „ó“. Intuitívny predpoklad, že dlhé samohlásky sú dlhšie vyslovované varianty svojich

základných tvarov bez dĺžňa, potvrdzuje okrem [tab 1] aj výzor ich vĺn. Na obrázkoch vzoriek „pip“ [obr 14] a „píp“ [obr 15] vidíme, že obe vlny „i“ aj „í“ pozostávajú z veľmi podobných opakujúcich sa segmentov vlny dĺžky asi 10 milisekúnd.



obr 14
vlna vzorky “pip”, podsvietené je „i“ (čas na vodorovnej osi je uvádzaný v sekundách)



obr 15
vlna vzorky “píp”, podsvietené je „í“ (čas na vodorovnej osi je uvádzaný v sekundách)

Tieto fakty spolu s tým, že pri rozpoznávaní čísiel nikdy nemusíme rozlišovať medzi krátkou a jej dlhou samohláskou, ma priviedli k tomu, že v ďalších súboroch foném som už zjednotil tieto podobne vyslovované vlny do jednej klasifikačnej kategórie. Z podobných dôvodov som zjednotil aj „ä“ s „e“ a „m“ s „n“, pred ktorými je fonéma „e“.

So vzorkami robenými podľa súboru rozpoznávaných zvukov 2 som previedol len niekoľko trénovaní neurónovej siete na jednom rečníkovi. Napriek 99% úspešnosti klasifikovania foném z trénovacej množiny vykazovalo rozpoznávanie čísiel neželané správanie. Napríklad príliš veľa rôznych okienok bolo nesprávne klasifikovaných ako niektorá z kategórií

obsahujúcich fonému „ť“ alebo „ť“. Predpokladal som, že to bolo spôsobené nevhodným spôsobom vyrábania vzoriek, keďže kategórie rozpoznávaných zvukov sa príliš vzájomne prelínali. Súbor foném 2 som teda zavrhol, ale nemám podložené že by to bol nevhodný spôsob označovania vzoriek. Na takéto tvrdenie by som potreboval urobiť viac testov a hlavne s väčším počtom vzoriek. K dispozícii som mal len 1 minútu pooznačovaných nahrávok[50].

6.2 Hlavný experiment

Hlavným experimentom tejto práce bolo maximalizovanie úspešnosti rozpoznávania čísiel nasledujúceho slovníka o veľkosti 16 klasifikovaných čísiel[51].

Základný slovník:

jeden	jede	
dva	dve	
tri		
štyri		
päť	pät	pä
šesť	šes	
sedem		
osem		
deväť	devät	devä
desať		
desiat		
štr		
násť		
cať		
sto	cto	
tisíc		

Okrem toho som sledoval aj úspešnosť rozpoznávania so slovníkom prirodzených čísiel od 1 po 999 999. Testovanými slovami boli čísla, ktoré boli použité na vzorky foném. Konkrétne: 15, 21, 44, 59, 205, 301, 916, 1011, 3793, 5814, 17810.

Úspešnosť som meral na ôsmich rečníkoch, ktorý zároveň nahrávali vzorky foném. Osobitne aj na jednom rečníkovi, z ktorého neboli robené vzorky[52]. Neurónová sieť využívala na trénovanie štyri nahrávky z každého z čísiel 15, 21, 44, 59, 205, 301, 916, 1011, 3793, 5814,

[50] Manuálne označiť minútu nahrávok trvá zhruba 8 hodín
[51] V slovníku je uvedených viac čísiel, pretože medzi slovami vyslovenými slovami v riadku nerozlišujeme.
[52] Pochopiteľne, úspešnosti merané na tomto jednom rečníkovi sú temer bezpredmetné

17810 od každého rečníka[53]. To bolo spolu zhruba 8 minút ručne pooznačovaných vzoriek foném. Približne 10% časových okienok bolo použitých ako validačná množina.

6.2.1 Hľadanie optimálnych parametrov

Celý systém rozpoznávača obsahuje desiatky parametrov, ktorých optimálna kombinácia sa dá v ich mnoho-rozmernom priestore „do nekonečna“ hľadať. Vhodné parametre pre skóre slova, ktoré som spomínal v časti 5.3.2 dokázali vylepšiť úspešnosť rozpoznávača o 30% a pomerne ľahko sa hľadali. Jednak pomocou niekoľko násobného spúšťania rozpoznávača na celej množine slov slovníka, a tiež analyzovaním výstupu Viterbiho dekodéra na niektorých problémových slovách. V dodatku 3 je príklad výstupu ANN a jeho spracovanie Viterbiho algoritmom.

Parametre a architektúru neurónovej siete spolu s vhodným výstupom fázy z frekvenčnej analýzy som hľadal pomocou osemdesiatich trénovaní ANN na vzorkách od jedného rečníka a štyridsiatich trénovaní na vzorkách od všetkých ôsmich rečníkov[54]. Väčšinou som testoval niekoľko trénovaní ANN na zistenie jedného parametra siete.

Najlepšie výsledky som dosiahol s nasledovnými parametrami:

Extrakcia pozorovaní zo zvukovej vlny: pre každé okienko s veľkosťou 5 milisekúnd (pričom vektory pozorovaní sa počítali na základe 12,5 milisekundovej oblasti) sa počítal dvanásť-rozmerný vektor MFCC koeficientov a energia signálu, aj s prvou a druhou deriváciou pre každý komponent vektora – spolu 39 rozmerný vektor pozorovaní.

Neurónová sieť: Skrytá vrstva obsahovala 120 neurónov, pričom som použil model TDNN s veľkosťou kontextu 5 pre skrytú vrstvu a 3 pre výstupnú vrstvu. Trénovanie: 200 iterácií s rýchlosťou učenia $3*10^{-5}$, a s momentovým faktorom 0,7. Rýchlosť učenia sa pre túto sieť násobila koeficientom 0,25 po každej epoche, keď sa úspešnosť na validačnej množine nezlepšila.

6.2.2 Úspešnosť rozpoznávania

Úspešnosť rozpoznávania s použitím ANN opísanej v predošlej časti je nasledovná: Klasifikovanie foném z trénovacej množiny: 86%, z validačnej množiny: 83,3%. Individuálne úspešnosti foném z validačnej množiny sú v nasledujúcej tabuľke:

[53] Snažil som sa o také zastúpenie foném v slovách, ktoré pomerne zodpovedá výskytu týchto foném vo všetkých číslach do milión. Zoznam slov vzoriek s výskytmi foném aj transripciou je v dodatku 2.
[54] Kompletná tabuľka parametrov a výsledkov trénovaní je na priloženom CD.

Fonéma	Počet testovaných okienok	Úspešnosť klasifikovania v [%]	Úspešnosť klasifikovania druhej úrovne v [%]
J	65	6,2	56,9
P	116	95,7	100
DX	354	87,6	94,1
D	70	60	84,3
DN	75	78,7	96
DV	190	77,9	95,3
V	61	68,9	88,5
R	99	78,8	92,9
T	499	66,3	84,6
TX	780	92,4	96,2
N	146	57,5	86,3
M	267	62,5	85,4
A	1112	96,3	97,8
E	987	79,8	95,1
I	560	91,8	98,4
IA	183	73,8	94,5
O	311	82,3	88,7
S	1349	95,5	97,5
SX	229	82,1	97,4
C	398	51	79,9

Klasifikačná tabuľka:

	J	P	DX	D	DN	DV	V	R	T	TX	N	M	A	E	I	IA	O	S	SX	C
J	6,2	0	1,5	1,5	1,5	0	0	0	0	0	0	0	0	37	49	0	0	3,1	0	0
P	0	96	0	0	0	4,3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DX	0	0	88	1,1	1,4	1,7	0	0	0,8	0,3	0	1,4	0	2	3,7	0	0	0	0	0
D	0	0	4,3	60	13	13	0	0	0	0	2,9	2,9	0	0	0	0	0	0	0	4,3
DN	0	0	12	1,3	79	0	0	0	2,7	0	5,3	0	0	0	0	0	0	0	0	0
DV	0	0	13	0,5	0	78	1,1	0	2,6	0	0,5	0	4,7	0	0	0	0	0	0	0
V	0	0	0	6,6	0	0	69	0	0	0	4,9	9,8	3,3	3,3	0	3,3	0	0	0	0
R	0	0	6,1	0	2	5,1	0	79	0	0	0	0	0	4	4	0	0	0	0	0
T	0	6,8	2,4	0,6	4,6	1,8	0	0	66	10	0	0	0	0	0	0	0	4,2	0	3,2
TX	0	0	0,1	0	0	0,3	0	0	4,9	92	0	0	0	0	0	0	0	0,8	0	1,5
N	0	0	0	0	18	3,4	0	0	0	0	58	16	4,1	1,4	0	0	0	0	0	0
M	0	0	0	0	18	0	1,5	0,4	0	0	3,7	63	7,9	6,4	0	0	0	0	0	0
A	0	0	0	0	0,6	0	0	0	0	0	1,2	0,5	96	1,3	0	0	0,1	0	0	0
E	0	0	0,3	0	0	0,4	0	0,3	0	0	0	0,1	10	80	5,1	3,9	0	0	0	0
I	0	0	0	0	0	0	0,7	0	0	0	0	0	0	7,5	92	0	0	0	0	0
IA	0	0	3,3	0	0	0	0	0	0	0	1,1	0	10	11	0,5	74	0	0	0	0
O	0	0	0	0	0	0,6	0	0	0	0	0	0,3	8,4	8,4	0	0	82	0	0	0
S	0	0,3	0	0	0	0	0	0	1	1,7	0	0	0	0	0	0	0	96	1,3	0,2
SX	0	0	0	0	0	0	0	0	0	3,1	0	0	0	0	0	0	0	14	82	0,9
C	0	0,3	5,3	2,5	3	1,3	0	0	5,5	3,8	0	0	0,3	0	0,5	0	0,3	26	0	51

Úspešnosti na slovách čísiel:

	Základný slovník (16 slov)	Slovník čísiel po milión-1, test na 11 číslach
Priemer na ôsmich trénovaných rečníkoch	90,63%	43,18%
1 konkrétny nezávislý rečník	81,25%	36,36%

Podstatnou hodnotou je rozpoznávanie na základnom slovníku pre ôsmich trénovaných rečníkov 90,63%. Môžeme ju považovať za priblíženie k hodnote na rozpoznávaní nezávislom od rečníka. Úspešnosti pre jedného netrénovaného rečníka len hovoria o potenciálnej sile ANN zovšeobecňovať, konkrétnu hodnotu 81,25 % pochopiteľne musíme považovať za orientačnú. Hodnoty na rozpoznávaní 11 čísiel do milión ukazujú, že aj napriek slušným výsledkom na slovách, z ktorých sa skladajú čísla do milión, nie je problém rozpoznávania na vetách z týchto slov uspokojivo riešiteľný.

7. Záver

V práci som dosiahol 90,63 % úspešnosť rozpoznávania na šestnástich slovách, z ktorých sa dajú skladať čísla do milión v slovenčine. Snažil som sa dopracovať k výsledkom, ktoré sa dosiahli v práci [MBA] na rozpoznávaní tridsaťjeden slov, z ktorých sa dajú skladať čísla anglického jazyka do sto. Tam sa dosiahol výsledok 91,7 %, pričom bolo k dispozícii 1821 rečníkov so štyrmi hodinami vzoriek na trénovanie a 618 testovacích rečníkov. Na proti tomu v tejto práci bolo k dispozícii osem rečníkov s ôsmimi minútami vzoriek, pričom úspešnosť som meral na tých istých rečníkoch.

V implementácii som využil techniky z rôznych oblastí umelej inteligencie: neurónové siete, skryté Markovové modely, rozpoznávanie izolovaných slov, ale aj poznatky z teórie pravdepodobnosti a teórie formálnych jazykov. Vylepšiť úspešnosť rozpoznávača by sa dala pravdepodobne zvolením ešte výhodnejšieho súboru rozpoznávaných zvukov, kde by sme zlúčili fonémy do kategórií, ktoré by boli optimálne pre rozpoznávanie slov čísiel. Nájdenie lepších parametrov systému rozpoznávača čísiel by sa dalo pomocou väčšieho množstva vzoriek a od viac rečníkov. S väčším množstvom vzoriek by sa dala určiť aj úspešnosť rozpoznávania čísiel s využitím neurónových sieti nezávisle od rečníka.

Dodatok 1

Súbor foném 1

graféma	fonéma	príklad	fonetický zápis	typ	použitá zvuková nahrávka	označenie klasifikovanej kategórie
a	a	papier	papI_^Er	samohláska	pap	a1
á	a:	pás	pa:s	samohláska	páp	a2
ä	{	mäso	m{sO	samohláska	päp	a3
c	ts	cena	tsEna	spoluhláska	aca	c1
d	d	voda	vOda	spoluhláska	ada	d1
ď	J\	háďa	h\a:J\a	spoluhláska	aďa	d2
e	E	pero	pErO	samohláska	pep	e1
i	I	pivo	pIvO	spoluhláska	pip	i1
í	I:	pískať	pI:skac	samohláska	píp	i2
ia	I_^a	piatok	pI_^atOk	dvojhláska	piap	ia
j	j	dvaja	dvaja	spoluhláska	aja	j1
l	l	skala	skala	spoluhláska	ala	l1
m	m	mama	mama	spoluhláska	ama	m1
n	n	rana, pánsky	rana, pa:nskI	spoluhláska	ana	n1
o	O	popol	pOpOl	samohláska	pop	o1
ó	O:	pól	pO:l	samohláska	póp	o2
p	p	popol	pOpOl	spoluhláska	apa	p1
r	r	para	para	spoluhláska	ara	r1
s	s	Osem	OsEm	spoluhláska	asa	s1
š	S	košeľa	koSELa	spoluhláska	aša	s2
t	t	vata	vata	spoluhláska	ata	t1
ť	c	platiť	placIc	spoluhláska	ať a	t2
u	U	puto	pUtO	samohláska	pup	u1
v	v	slovo	slOvO	spoluhláska	ava	v1

Transkripcia čísiel pomocou súboru foném 1:

1	jeden	jEdEn	j	E	d	E	n				
1	jedna	jEdna	j	E	d	n	a				
2	dva	dva	d	v	a						
2	dve	dvE	d	v	E						
3	tri	trI	t	r	I						
4	štyri	StIrI	S	t	I	r	I				
5	päť	p{c	p	{	t						
6	šesť	SEsc	S	E	s	c					
7	sedem	sEJ\Em	s	E	J\	E	m				
8	osem	osEm	o	s	E	m					
9	deväť	J\Ev{c	J\	E	v	{	c				
10	desať	J\Esac	J\	E	s	a	c				
11	jedenásť	jEdEna:sc	j	E	d	E	n	a:	s	c	
12	dvanásť	dvana:sc	d	v	a	n	a:	s	c		
13	trinásť	trIna:sc	t	r	I	n	a:	s	c		
14	štrnásť	Strna:sc	S	t	r	n	a:	s	c		
15	pätnásť	p{tna:sc	p	{	t	n	a:	s	c		
16	šestnásť	SEsna:sc	S	E	s	n	a:	s	c		
17	sedemnásť	sEJ\Emna:sc	S	E	J\	E	m	n	a:	s	c
18	osemnásť	osEmna:sc	o	s	E	m	n	a:	s	c	
19	deväťnásť	J\Ev{tna:sc	J\	E	v	{	t	n	a:	s	c
20	dvadsať	dvatsac	d	v	a	ts	a	c			
30	tridsať	tritsac	t	r	i	ts	a	c			
40	štyridsať	StIrItsac	S	t	I	r	I	ts	a	c	
50	päťdesiat	p{J\EsI_^at	p	{	J\	E	s	I_^a	t		
60	šesťdesiat	SEsJ\EsI_^at	S	E	s	J\	E	s	I_^a	t	
70	sedemdesiat	sEJ\EmJ\EsI_^at	s	E	J\	E	m	J\	E	s	I_^a
80	osemdesiat	osEmJ\EsI_^at	o	s	E	m	J\	E	s	I_^a	t
90	deväťdesiat	J\Ev{J\EsI_^at	J\	E	v	{	J\	E	s	I_^a	t
100	sto	stO	s	t	O						
1000	tisíc	cIsI:ts	c	I	s	I:	ts				
1000000	milión	mIIIO:n	m	I	I	I	O:	n			
1000000	miliónov	mIIIO:nOU	m	I	I	I	O:	n	O	U	
1000000	milióny	mIIIO:nI	m	I	I	I	O:	n	I		
1000000000	miliarda	mIII_^arda	m	I	I	I_^a	r	d	a		
1000000000	miliardy	mIII_^ardI	m	I	I	I_^a	r	d	I		
1000000000	miliárd	mIIIa:rd	m	I	I	I	a:	r	d		

Súbor foném 2:

je-	Je	J	3	jedna	jeden	jede			
pe-	Pe	P	3	peť	pet	pe			
ďe-	DXe	D	5	ďeveť	ďevet	ďeve	ďesať	ďesiat	
-eďe-	eDXe	D	1	sedem					
-ede-	eDe	D	2	jeden	jede				
-aď	aDX	D	1	caď					
-edna-	eDNa	D,N	1	jedna					
dva-	DVa	D,V	1	dva					
dve-	DVe	D,V	1	dve					
-eve-	eVe	V	3	ďeveť	ďevet	ďeve			
-iri-	iRi	R	1	štiri					
tri-	TRi	T,R	1	tri					
-štr	sxTR	T,R	1	štr					
-šti	sxTi	T	1	štiri					
-sť	sTX	T	1	šesť	násť	tnásť			
-eť	eTX	T	2	peť	ďeveť				
-et	eT	T	2	pet	ďevet				
-sto-	sTo	T	1	sto					
-cto-	CTo	C,T	1	cto					
-ať	aTX	T	2	ďesať	cať				
-iat	iaT	T	1	ďesiat					
ťi-	TXi	T	1	ťisíc					
tná-	TNá	T,N	1	tnásť					
ná-	Na	N	1	násť					
-em	eM	M	2	sedem	osem				
-en	eN	N	1	jeden					
-íc	ixC	C	1	ťisíc					
ca-	Ca	C	1	cať					
-a-	A		7	jedna	dva	ďesať	násť	tnásť	cať
-e-	E		17	jedna	jeden	jede	dve	peť	...
-i-	I		5	tri	štiri	ťisíc			
-ia-	IA		1	ďesiat					
-o-	O		3	osem	sto	cto			
-s-	S		10	šesť	šes	sedem	osem	ďesať	...
-š-	SX		4	štiri	štr	šesť	šes		

Transkripcia do súboru foném 2:

jedna	pau	Je	E	eDNa	A	pau	
jeden	pau	Je	E	eDe	E	eN	pau
jede	pau	Je	E	eDe	E	pau	
dva	pau	DVa	A	pau			
dve	pau	DVe	E	pau			
tri	pau	TRi	I	pau			
štiri	pau	SX	sxTi	I	iRi	I	pau
štr	pau	SX	sxTR	pau			
peť	pau	Pe	E	eTX	pau		
pet	pau	Pe	E	eT	pau		
pe	pau	Pe	E	pau			
šesť	pau	SX	E	S	sTX	pau	
šes	pau	SX	E	S	pau		
sedem	pau	S	E	eDXe	E	eM	pau
osem	pau	O	S	E	eM	pau	
ďeveť	pau	DXe	E	eVe	E	eTX	pau
ďevet	pau	DXe	E	eVe	E	eT	pau
ďeve	pau	DXe	E	eVe	E	pau	
ďesať	pau	DXe	E	S	A	aTX	pau
násť	pau	Na	A	S	sTX	pau	
tnásť	pau	TNa	A	S	sTX	pau	
cať	pau	Ca	A	aTX	pau		
caď	pau	Ca	A	aDX	pau		
ďesiat	pau	DXe	E	S	IA	iaT	pau
sto	pau	S	sTo	O	pau		
cto	pau	CTo	O	pau			
ťisíc	pau	TXi	I	S	I	iC	pau

Dodatok 2

Súbor foném 3 - použitý v hlavnom experimente spolu s výskytmi kategórií foném v trénovaných slovách:

				sum	**21**	**301**	**1011**	**205**	**3793**	**44**	**5814**	**15**	**59**	**916**	**17810**	**10**
je-	**1**	J	J	3	1	1	1									
pe-	**2**	P	P	4				1			1	1	1			
ďe-, -eďe-, -aď	**3**	DX	Ď	10	1				3				2	1	2	1
-ede-	**4**	D	D	2		1	1									
-edna-	**5**	DN	DN	1	1											
dva-, dve-	**6**	DV	DV	2	1			1								
-eve-	**7**	V	V	3					1				1	1		
-iri-, tri-, -štr	**8**	R	R	6		1			2	2	1					
tri-, -štr, ti-, -t, to-	**9**	T	T	14		2		1	3	2	2	1	1	1	1	
ťi-, -ť	**10**	TX	Ť	13			2	1	1	1	2	1	1	1	2	1
ná-	**11**	N	N	5			1				1	1		1	1	
-em, -en	**12**	M	M,N	5		1			1		1				2	
-a-	**13**	A	A	11	3		1			1	1	1		1	2	1
-e-	**14**	E	E	27	1	2	2	2	5		2	1	4	3	4	1
-i-	**15**	I	I	15		1	2		4	4	2				2	
-ia-	**16**	IA	IA	2					1				1			
-o-	**17**	O	O	8		1		1	1		2			1	2	
-s-	**18**	S	S	23		1	2	1	4		4	1	1	2	6	1
-š-	**19**	SX	SX	4						2	1			1		
-c-	**20**	C	C	7	1		1		1	1	1			1	1	
					9	11	13	8	27	13	21	7	12	14	25	5

Transkripcia rozpoznávaných čísiel použitá v hlavnom experimente:

jedna	pau	J	E	DN	A	pau	
jeden	pau	J	E	D	E	M	pau
jede	pau	J	E	D	E	pau	
dva	pau	DV	A	pau			
dve	pau	DV	E	pau			
tri	pau	T	R	I	pau		
štiri	pau	SX	T	I	iRi	I	pau
štr	pau	SX	T	R	pau		
peť	pau	P	E	pau	TX	pau	
pet	pau	P	E	pau	T	pau	
pe	pau	P	E	pau			
šesť	pau	SX	E	S	TX	pau	
šes	pau	SX	E	S	pau		
sedem	pau	S	E	DX	E	M	pau
osem	pau	O	S	E	M	pau	
ďeveť	pau	DX	E	V	E	TX	pau
ďevet	pau	DX	E	V	E	T	pau
ďeve	pau	DX	E	V	E	pau	
ďesať	pau	DX	E	S	A	TX	pau
násť	pau	Na	A	S	TX	pau	
násť	pau	Na	A	S	pau		
cať	pau	C	A	TX	pau		
caď	pau	C	A	DX	pau		
cať	pau	C	A	pau			
ďesiat	pau	DX	E	S	IA	T	pau
ďesiat	pau	DX	E	S	IA	pau	
sto	pau	S	T	O	pau		
cto	pau	C	T	O	pau		
ťisíc	pau	TX	I	S	I	C	pau

Dodatok 3

K IMPLEMNTÁCII:

Celý systém je prepojením toolkitov HTK (pre frekvenčnú analýzu) a NICO (vytváranie neurónových sietí) a modulu viterbi v ktorom sú kľúčové funkcie viterbi a recognize_word. Práca je naprogramovaná v jazyku C v ANSI norme, pre kompilátor GCC. Testovaná bola na systéme Linux Fedora Core 5 pre 32 aj 64 bitovú verziu.

Na priloženom CD sú v adresári TESTY dávkové súbory testing*.sh, kde je treba pre spustenie zadať poradové číslo trénovania. Pre spustenie rozpoznávania je potrebný súbor s natrénovanou ANN, tie sa nachádzajú v adresári Trenovania ANN. Tiež konfiguračný súbor pre HTK **hrest.conf**. Okrem nich aj súbory s maticami prechodov a topológiami slov s príponami .**tran** a .**topo**. Vygeneruje ich modul **generateTransition**, ktorý potrebuje súbor so zoznamom rozpoznávaných slov a jednotlivé labelovacie súbory s koncovkou .**lab**. Samotné rozpoznávanie slova sa vykonáva pomocou modulu **viterbi**, ten využíva výstup ANN, ktorý je v súbore s koncovkou .**act,** potrebuje aj súbor s maticou prechodov phoneme.tran a zoznam rozpoznávaných slov so stavmi **state_word.txt. makeSample** generuje dávkový súbor na rozpoznávanie zvuku uloženého vo formáte .**wav**, **xspeaker** generuje dávkový súbor pomocou ktorého sa trénuje ANN, využíva pritom nastavenia v **phoneme.c** a templaty **hrest_conf.template** pre konfiguračný súbor HTK a **train.template** na definovanie ANN pre modul Niko.

Všetky moduly z adresára LIB sa skompilujú pomocou dávkového súboru **make.sh**. Voliteľné parametre sa dajú pred skompilovaním meniť v súbore **phoneme.c**

HTK toolkit aj s dokumentáciou je voľne stiahnuteľný na http://htk.eng.cam.ac.uk/

NIKO toolkit aj s dokumentáciou je voľne stiahnuteľný na http://nico.nikkostrom.com/

Literatúra

[UNS] Vladimír Kvasnička, Ľubica Beňušková, Jiří Pospíchal, Igor Farkaš, Peter Tiňo, Andrej Kráľ -"Úvod do teórie neurónových sietí", IRIS 1997

[PSU] Josef Psutka - "Komunikace s poèítaèem mluvenou øeèí", ACADEMIA PRAHA 1995

diplomové práce:

[NAG] Marek Nagy - „*Experimentálny systém rozpoznávania reči*“ MFF-UK 1999

[OND] Jaroslav Ondriska - „*Porovnanie úspešnosti rozpoznávania izlovaných slov rôznymi prístupmi pri telefónnej kvalite mobilných telefónov*“,FMFI-UK 2002

[SEM] Ondrej Seman - „*Ovládanie počítača hlasom*“, FMFI-UK 2004

články:

[CSL] John-Paul Hosom, Ron Cole, Mark Fanty -„*Speech Recognition Using Neural Networks*“, Center for Spoken Language Understanding -Oregon Graduate Institute of Science and Technology 1999

[MBA] Naghmeh Nikki Mirghafori -„*A Multi Band Approach to Automatic Speech Recognition*“, 1999

[ASR] Lilia Lazli, Mokhtar Sellami „*Connectionist Probability Estimators in HMM Arabic Speech Recognition Using Fuzzy Logic*“, Department of Computer Science at Faculty of Engineer Science -Badji Mokhtar University 2003

[RAB] Lawrence R. Rabiner „*A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*” 1989

[TSR] DongSuk Yuk, James Flanagan „*Telephone speech recognition using neural networks and HMM*“, Department of Computer Science, Rutgers Univesity, New Brunswick, NJ

Prednášky:

[HMM] John-Paul Hosom „*Hidden Markov Models for Speech Recognition*”, Department of Computer Science and Engineering OGI School of Science & Engineering OHSU, 2006

[NIK] Nikko Strom - *The NICO Toolkit Manual* http://nico.nikkostrom.com/doc/

[HTK] *HTK Speech Recognition Toolkit*, http://htk.eng.cam.ac.uk

[IVA] SAMPA for Slovak, http://www.ivanecky.sk/SAMPA/slovak.html